(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－237808
（P2002－237808A）
(43)公開日　平成14年８月23日(2002.8.23)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード(参考)
Ｈ０４Ｌ　9/08		Ｇ０６Ｆ　17/60	１４２	５Ｊ１０４
Ｇ０６Ｆ　17/60	１４２	Ｈ０４Ｌ　9/00	６０１Ａ	

審査請求　未請求　請求項の数27　ＯＬ　（全 42 頁）

(21)出願番号　特願2001－33915(P2001－33915)

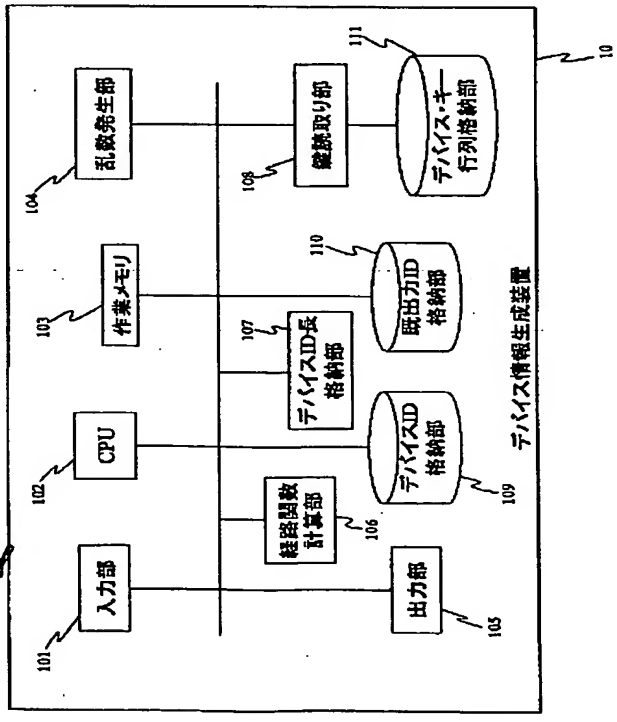
(22)出願日　平成13年２月９日(2001.2.9)

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(72)発明者　上林　遼
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(74)代理人　100083161
弁理士　外川　英明

Ｆターム(参考)　5J104 AA12 AA16 EA06 EA17 NA02

(54)【発明の名称】　デバイス情報生成装置、デバイス情報生成方法、制御データ生成装置、制御データ生成方法、情報利用装置、情報利用方法、および記憶媒体

(57)【要約】
【課題】　利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避するデバイス情報生成装置を提供する。
【解決手段】　利用デバイス３０に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、計算する経路関数計算部１０６を備えた。



【特許請求の範囲】

【請求項１】 情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成する生成手段とを具備することを特徴とするデバイス情報生成装置。

【請求項２】 情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を生成する第１生成手段と、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成する第２生成手段とを具備することを特徴とするデバイス情報生成装置。

【請求項３】 デバイス情報を生成するデバイス情報生成装置であって、数字の並びであるデバイス識別子を構成する数字の一部または全部の数字の並びである、部分経路の少なくとも一つに関して、当該部分経路における経路関数値を計算する、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項４】 請求項３のデバイス情報生成装置であって、互いに相異なる入力経路に対して互いに相異なる経路関数値を対応させる、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項５】 請求項1のデバイス情報生成装置であって、数字の並びである経路に対応して定まる、デバイス・キー行列の複数の成分に依存する経路関数値を出力する、経路関数計算部を具備する事を特徴とする、デバイス情報生成装置。

【請求項６】 制御データを生成する制御データ生成装置であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力する、随伴頂点集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項７】 制御データを生成する制御データ生成装置であって、経路を少なくとも一つ設定することができ、入力された経路が、設定された経路に属する数字の一部または全部を用いて作られる部分経路に一致するか否かを判定し、判定結果を出力する随伴頂点集合判定部を具備する事を特徴とする、制御データ生成装置。

【請求項８】 制御データを生成する制御データ生成装置であって、経路を少なくとも一つ設定することができ、入力された経路が、設定された経路に少なくとも一つの数字を追加して作られる経路に一致するか否かを判定し、判定結果を出力する随伴頂点集合判定部を具備する事を特徴とする、制御データ生成装置。

【請求項９】 制御データを生成する制御データ生成装置であって、入力された経路に少なくとも一つの数字を追加して作られる経路を、少なくとも一つ出力する、随伴頂点集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項１０】 制御データを生成する制御データ生成装置であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路について、当該部分経路に属する数字の一部または全部を変更して作られる経路を少なくとも一つ出力する、境界集合計算部を具備する事を特徴とする、制御データ生成装置。

【請求項１１】 制御データを生成する制御データ生成装置であって、入力された経路に対して定まる経路関数値を計算する、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項１２】 請求項１１の制御データ生成装置であって、互いに相異なる入力経路に対して互いに相異なる経路関数値を対応させる、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項１３】 請求項１１の制御データ生成装置であって、入力された経路に対して定まる、デバイス・キー行列の複数の成分に依存する経路関数値を出力する、経路関数計算部を具備する事を特徴とする、制御データ生成装置。

【請求項１４】 制御データを生成する制御データ生成装置であって、経路と当該経路に対応付けられた数値の組を、少なくとも一組出力する事を特徴とする、制御データ生成装置。

【請求項１５】 請求項１４の制御データ生成装置であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路について、当該部分経路に属する数字の一部または全部を変更して作られる経路を少なくとも一つ出力する、境界集合計算部を具備し、出力される経路が当該境界集合計算部の出力である事を特徴とする、制御データ生成装置。

【請求項１６】 請求項１１、１２または１３のいずれかの制御データ生成装置であって、当該制御データ生成装置は、経路と当該経路に対応付けられた数値の組を少なくとも一組出力し、経路に対応づけられた数値は、当該経路の入力に対する経路関数計算部の出力値を鍵として、データ利用に必要な情報であるマスター鍵を暗号化する事によって得られる、リボケーション関数値である事を特徴とする、制御データ生成装置。

【請求項１７】 請求項６乃至１４の何れかの制御データ生成装置が出力した制御データを含む事を特徴とする、記録媒体。

【請求項１８】 データ利用を行う情報利用装置であって、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値が、少なくとも一つ予め格納されている、デバイス情報格納部を具備する事を特徴とする、情報利用装置。

【請求項１９】 データ利用を行う情報利用装置であって、請求項１乃至４何れかのデバイス情報生成装置が出力したデバイス情報を予め格納するデバイス情報格納部を具備する事を特徴とする、情報利用装置。

【請求項２０】 データ利用を行う情報利用装置であって、経路と当該経路に対応付けられた数値の組を少なく

とも一組含む制御データを読みこむ、制御データ入力部を具備する事を特徴とする、情報利用装置。

【請求項２１】　データ利用を行う情報利用装置であって、請求項６乃至１６の何れかの制御データ生成装置が出力した制御データを読みこむ、制御データ入力部を具備する事を特徴とする、情報利用装置。

【請求項２２】　請求項１８乃至２１記載の情報利用装置において、入力された制御データの中から、デバイス識別子の一部または全部の数字を並べて得られる、部分経路に一致する経路を検索する事を特徴とする、情報利用装置。

【請求項２３】　請求項２２記載の情報利用装置であって、入力された制御データの中から、デバイスIDの一部または全部の数字を並べて得られる、部分経路に一致する経路を検索し、制御データ中に、そのような経路が見つからなかった場合、当該利用デバイスがリボークされていると判断し、予め定められたリボーク処理を行う事を特徴とする、情報利用装置。

【請求項２４】　請求項２２の情報利用装置であって、入力された制御データの中から、デバイスＩＤの一部または全部の数字を並べて得られる、部分経路に一致する経路を検索し、制御データ中に、そのような経路が見つかった場合、制御データから当該経路に対応するリボケーション関数値を読み出し、またデバイス情報格納部から当該経路に対応する経路関数値を読み出し、当該経路関数値を鍵としてリボケーション関数値を復号する事を特徴とする、情報利用装置。

【請求項２５】　情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成することを特徴とするデバイス情報生成方法。

【請求項２６】　制御データを生成する制御データ生成方法であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力することを特徴とする制御データ生成方法。

【請求項２７】　データ利用を行う情報利用装置に、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値の少なくとも一つを格納することを特徴とする情報利用方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、電子化データの情報利用装置に関わる。また、本発明は、当該利用デバイスにおける電子化データの利用を制御する制御データを作成する、制御データ作成装置に関わる。また、デバイス情報生成装置に関わる。更に、本発明は、当該制御データが記録されたデータ記録媒体に関わる。

【０００２】

【従来の技術】利用デバイスにおける電子化データ（以後単にデータと言う）の利用を、制御データによって制御する技術として、リボケーションと呼ばれる技術が知られている。リボケーションは、しばしば電子化著作物の著作権保護に応用されている。リボケーションの概念を以下に説明する。

【０００３】利用デバイスは、予め固有の情報（デバイス情報）を割り当てられており、デバイス内に保持している。デバイス情報は、個々のデバイス毎に異なっている。あるいは、一群のデバイスが同一のデバイス情報を割り当てられていても良い。例えば、Ａ社が製造したＣＤ再生デバイスは全て同一のデバイス情報を持つ、と言うような場合が、それに当たる。

【０００４】今、ある利用デバイスにおいて、セキュリティ対策が不十分であった等の理由により、データの不正利用が発生したと仮定する。この場合、不正利用が発生したデバイスにおけるデータ利用を制限／禁止して、被害の拡大を防ぐ事が重要である。

【０００５】このため、制御データは、利用デバイスへ配布され、この制御データを利用デバイスで読み取り、当該制御データと、当該利用デバイスが保持するデバイス情報とに基づいて、データ利用の可否を決定する。これにより、少なくとも不正利用が発生した利用デバイスを含む、一群のデバイスはデータの利用が制限／禁止される。制御データは、例えば、放送やネットワークを通じてデータを送信する場合には、データに付随させて利用端末に配布したり、ＣＤ－ＲＯＭやＤＶＤ－ＲＯＭ等のパッケージ・メディアを通じて流通する場合には、パッケージ・メディアに記録したりして配布される。

【０００６】このように、１）利用デバイスに対して制御データが配布され、２）利用デバイスは当該制御データを読み込み、３）利用デバイスは、デバイス情報と制御データに基づいて、データ利用の可否を判断するといった一連の処理により、利用デバイスにおけるデータ利用を制限／禁止する事を、リボケーションと言う。利用デバイスにおけるデータ利用がリボケーションによって制限／禁止されたとき、当該利用デバイスはリボークされたと言う。制御データの配布は、リボケーションを管理する主体（通常は法人）が行う。これをリボーク主体と呼ぶ。

【０００７】リボケーションにおいて、データ利用制限／禁止の対象となる利用デバイスを（リボーク）対象デバイスと呼ぶ。リボケーションは、デバイス情報に基づいて行われるため、同一のデバイス情報を持つデバイスは（例え複数存在しても）区別されない。しかし、後述のＭＫＢ技術において見られるように、対象デバイスとデバイス情報が異なっている利用デバイスであっても、データ利用が制限／禁止される現象が発生する事がある。この現象を、リボケーションの錯誤と呼ぶ事にする。これはリボケーションの副作用であり、利用デバイスを使用する消費者の便宜を著しく損なう。この種の副作用の発生を極力避ける事は、リボケーション技術にと

って非常に重要である。

【０００８】代表的なリボケーション技術の一つとして、リボケーション・リスト（ＲＬ）の技術が良く知られている。ＲＬは、制御データであり、通常はデータに付随した形で利用デバイスに配布される。ＲＬには、当該データの利用を禁止するデバイスのデバイス情報が列挙されている。利用デバイスは、データ利用に先立ってＲＬを読取り、自分自身のデバイス情報がＲＬに含まれているか否かをチェックする。デバイス情報がＲＬに含まれない場合、当該利用デバイスはデータを利用する。一方、デバイス情報がＲＬに含まれる場合、当該利用デバイスはデータ利用を行わない。ＲＬの改ざんを防ぐ為、ＲＬを暗号化する事がしばしば行われる。

【０００９】ＲＬ技術では、対象デバイスのデバイス情報を個別に指定する事ができる。また、リボケーションの錯誤は発生しない。ところが、ＲＬ技術は、セキュリティ的に看過し難い問題点を有している。リボーク対象デバイスがデータ利用を中止するのは、当該利用デバイスが自分自身のデバイス情報をＲＬ中に見出したが故に、データ利用を言わば「自粛」するからに過ぎない。データ利用に必要な情報の入手は、リボケーションの判断とは独立に行われる。そもそも、リボーク対象デバイスとは、改造等が施された可能性があり、「信用できない」利用デバイスである。リボーク対象デバイスが、データ利用を「自粛」しないよう改造されていれば、ＲＬのリボケーションは全く効果を持たない。

【００１０】メディア・キー・ブロック（ＭＫＢ）の技術は、ＲＬ技術が持つ上記のセキュリティ上の問題点を解決する、リボケーション技術の一つである。ＭＫＢ技術では、先ず、デバイス・キー行列KDを用意する。KDの行、列の数をそれぞれm,nとおく。またKDのi行j列成分をk[i,j]（ただし、0 <= i < n, 0 <= j < n）と表記する。KDの各成分は、例えば、乱数発生器で生成した乱数である。マスター鍵をKとする。マスター鍵は、データ利用に必要な情報の一つである。例えば、データがデータ・キーで暗号化されており、マスター鍵で暗号化されたデータ・キーが、データと共に利用デバイスに配布されている。マスター鍵を得た利用デバイスは、これを用いてデータ・キーを復号し、更にデータを復号して利用する事ができる。

【００１１】上記のデバイス・キー行列KDに対して、デバイス情報は、写像p:[0, ..., n- 1]→[0, ..., m - 1]と、pが誘導するデバイス鍵の列：KD(p) = [k[p(0), 0], k[p(1), 1], ..., k[p(n - 1), n - 1]]の組(p, KD(p))とである。また、制御データは、m行n列の行列Mである。MをＭＫＢと呼ぶ。Mのi行j列成分をM[i, j]と表記する。Mの初期値（即ち、リボケーションがないときの値）は、M[i, j] = Enc(k[i, j], K)で与えられる。Encは、適当な暗号化アルゴリズムによる暗号化関数である。鍵wによってデータxを暗号化した結果をEnc(w, x)と表記する。

【００１２】デバイス情報(p, KD(p))を有する利用デバイスは、ＭＫＢMを読取り、図３９に示す処理を行なう。この処理をＭＫＢ処理と呼ぶ。ＭＫＢ処理において、Decは、暗号化関数Encに対応する復号関数である。Dec(w, x)は、データxを鍵wで復号した結果を示す。定義から明らかに、Dec(w, Enc(w, x)) = xとなる。

【００１３】また、nullは予約された特別な数値である。特にnull≠Kでなければならない。p及びKD(p)は、それぞれ配列P及びKDPに格納されているものとする。P[j]= p(j)、KDP[j] = k[p(j), j]と示す。

【００１４】NNumは多倍長整数のクラスである。Mの初期値に対して、リボーク処理が、デバイス情報(p, KD(p))に関わりなくResult = Kを与える事は容易に読取れる。

【００１５】今、デバイス情報(a, KD(a))を有する利用デバイスDをリボークする事を考える。この時、リボーク主体は、次のＭＫＢ M'を利用デバイスに配布する。

M'[a(j), j] = Enc(k[a[j], j], null)
M'[i, j] = Enc(k[i, j], K) if i ≠ a(j)

明らかに、利用デバイスDがＭＫＢ M'を処理して得る結果はnullであり、マスター鍵Kを得る事はできない。一方、D以外の利用機器D'がM'を処理すれば、Kを得る。利用機器Dは、しかもDだけ、マスター鍵Kを得る事ができない。その結果、利用機器Dはデータ復号等の処理に進むことができず、リボークされた事になる。

【００１６】ＭＫＢ技術において、ＭＫＢ処理を実行するのは利用機器である。しかし、ＲＬの場合とは異なり、リボーク処理を行なうか否かの判断は利用機器に依存していない。ある利用機器がメディア・キー・ブロックによってリボークされている場合、当該利用機器に割り当てられているデバイス情報をどのように利用しても、マスター鍵を取り出す事はできない。従って、ＭＫＢ技術は、ＲＬが持つ先述のセキュリティ上の問題点を解決している。

【００１７】ＭＫＢ技術には、しかし、重大な欠点が存在する。即ち、複数の利用機器がリボークされた場合、リボケーションの錯誤が発生する可能性がある。簡単の為に小さなＭＫＢを用いて、この事を説明する。デバイス・キー行列のサイズを4行4列とし、ＭＫＢ Mの成分が次の様に与えられているとする。

M[2, 0] = Enc(k[2, 0], null), M[2, 1] = Enc(k[2, 1], null),
M[1, 2] = Enc(k[1, 2], null), M[3, 3] = Enc(k[3, 3], null),
M[i, j] = Enc(k[i, j], K) （上記以外）

上記MKBでは、デバイス情報(p, KD[p])を持つ利用デバイスD2が（そして、D2だけが）リボークされている。ただし、p = [2, 2, 1, 3]。

【００１８】ここで更に、デバイス情報(p', KD[p'])によって指定される利用デバイスD3をリボークする必要が生じたとする。ただし、p' = [1, 3, 1, 2]。ＭＫＢMは更新され、M'となる。M'の各成分は次の様に与えられる：

M'[2, 0] = M[2, 0], M'[1, 0] = Enc(k[1, 0], null),
M'[2, 1] = M[2, 1], M'[3, 1] = Enc(k[3, 1], null),
M'[1, 2] = M[1, 2],
M'[3, 3] = M[3, 3], M'[2, 3] = Enc(k[2, 3], null)
M'[i, j] = Enc(k[i, j], K)　　（上記以外）

M'によって、利用デバイスD2及びD3は確かにリボークされる。しかし、同時に、例えばデバイス情報(p'', KD[p''])を有する利用デバイスまでリボークされている。ただし、p'' = [2, 3, 1, 3]。D2、D3以外にはからずもリボークされてしまう利用デバイスは、全部で6台ある。

【００１９】ＭＫＢ技術においては、一般に、s台の利用デバイス（s個のデバイス情報）をリボークした場合、最大sn - s台の利用デバイスが、リボケーションの錯誤によってリボークされる。「無実の罪」を着せられた利用デバイスのユーザーも、リボーク対象デバイスのユーザーと共に、データ利用を制限される不都合を被る事となる。場合によっては、この事が重大な経済的損失等に繋がる可能性も否定できない。ＭＫＢをリボケーション技術として採用した場合、利用デバイスの供給業者や製造業者は、ユーザーからの苦情や損害賠償の要求など、潜在的な製品トラブルを抱える事になると言っても過言ではない。

【００２０】ＭＫＢ技術において、ある利用デバイスがリボケーションの錯誤によってリボークされる確率は、一般的な仮定の下で、リボーク対象デバイスの数に対して指数的に増大する。この事は、実際には、デバイス情報全体の中で極く少数のデバイス情報しかリボークできないと言う事を意味している。利用デバイス供給業者或いは製造業者も、錯誤の数が少数に止まっている間は、苦情に個別に対応しトラブルを処理する事が可能である。

【００２１】MKB技術を用いたリボケーションに必要な装置と、その構成及び動作を、以下若干詳細に述べる。これは、ＭＫＢ技術と本発明との相違を明確に述べる為である。ＭＫＢ技術におけるデバイス情報生成装置５０の構成を、図４０に示す。また、その動作を、図４１及び図４２に示す。デバイス・キー行列KDは二次元の配列として、デバイス・キー格納部５０９に格納されている。

k[0, 0] k[0, 1] k[0, 2]
k[1, 0] k[1, 1] k[1, 2]

デバイス･キー行列自体は、乱数発生器を用いるなど適当な方法により、予め作成されているものとする。乱数発生部５０４は、乱数発生指示を受け取ると、0または1の値をとる乱数を3個発生する。鍵読取り部５０８は、乱数を受け取り、当該乱数を行番号と見なして、デバイス･キー行列の各列から順に、当該行番号によって指定される要素を読みとる。

【００２２】機器情報格納部５０６には、次の二つの情報が記憶される。

行位置の並び：[R0, R1, R2]
鍵の並び：[k[R0, 0], k[R1, 1], k[R2, 2]]

これらが、機器情報を構成する。既出力情報格納部５０７には、行位置の並びが追加書きで記録される。従って、既出力情報格納部５０７には、出力された行位置の並びが全て記録されている。

【００２３】次にＭＫＢ生成装置の構成を図４３に示す。図４４及び図４５にＭＫＢ生成時の動作を示す。先の機器情報生成装置５０の場合と同様、デバイス･キー行列は、先述のデバイス・キー行列生成装置を用いて既に生成されていて、下記の二次元配列として、デバイス･キー行列格納部６１２に格納されているものとする。

K[0, 0] k[0, 1] k[0, 2]
K[1, 0] k[1, 1] k[1, 2]

鍵読取り部６１１は、二つの引数を受け取る。二つの引数は、デバイス･キー行列の行と列の番号であり、鍵読取り部６１１は、これらの番号によって指定されるデバイス･キー行列の要素を返す。ＭＫＢは、二次元の配列として、メディア・キー・ブロック格納部６１０に格納される。

M[0, 0] M[0, 1] M[0, 2]
M[1, 0] M[1, 1] M[1 ,2]

なお、図４４及び図４５において、i, jは配列の添字を格納する為の変数であり、例えば、ＣＰＵ６０５内のメモリに割り当てられる。

【００２４】図４６はＭＫＢ更新時のＭＫＢ生成装置６０の動作である。リボークすべき利用機器の機器情報を指定する必要があるが、それはＭＫＢの各列の行番号で指定される。例えば、リボークすべき利用機器のデバイス情報((1, 0, 0), KD(1, 0, 0))とする。この場合、ＭＫＢ生成装置６０のリボーク情報入力部６０２に、次のデータを入力して指定する。

(1(0), 1(1), 1(2)) = (1, 0, 0).

ＣＰＵ６０５は、これを順に零列目、一列目、及び二列目の行番号と見なし、それぞれをＭＫＢの配列要素の指定、即ち行番号と列番号のペア、に変換して更新部６０

９に順次入力する。更新部６０９は入力された、当該行番号と列番号のペアによって指定されるメディア・キー・ブロックの要素を無効化する。

【００２５】ＭＫＢに基づくリボケーション方式に準拠する利用機器７０の構成を図４７に示す。また、その動作を図４８及び図４９に示す。利用機器７０のＭＫＢ入力部７０１は、次のＭＫＢを読み込む。

M[0, 0] M[0, 1] M[0, 2]
M[1, 0] M[1, 1] M[1, 2]

読み込まれたＭＫＢは、ＭＫＢ格納部７０２に格納される。デバイス情報格納部７０５は、例えば、機器情報(1, KD(1))を格納している。なお、KD(1)の各成分を、KD(1) = (k(0), k(1), k(2))のように表記する。

【００２６】ＣＰＵ７０３は、デバイス情報格納部７０５からデータを順に読取り、ＭＫＢに適用して行く。即ち、先ず変数jの値を1として、デバイス情報格納部７０５からl(j)を読取り、配列要素読取り部７０７に数の組(l(j), j)を送る。配列要素読取り部７０７は、ＭＫＢ格納部７０２から要素M[l(j), j]を読み出してＣＰＵ７０３に返す。ＣＰＵ７０３は、M[l(j), j]を復号部７０８に送る。

【００２７】次いで、機器情報からk(j)を読取り、それを復号部７０８に送る。復号部７０８は、鍵k(j)でM[l(j), j]を復号し、その結果をＣＰＵ７０３に返す。ＣＰＵ７０３は、当該復号結果を、変数Resultに一時的に格納する。変数Resultの値がnullに等しければ、ＣＰＵ７０３はjを1増やす。j < 3なら、ＣＰＵ７０３は上記動作を繰り返す。さもなければ、ＣＰＵ７０３はメディア・キー・ブロック処理動作を停止する。

【００２８】j = 0, 1, 2のいずれかに対して、変数Resultの値がnullと異なっていた場合、CPUはResultの値を、マスター鍵として、データ利用部７０９に送る。次いで、ＣＰＵ７０３はデータ入力部７０６からデータを読取り、当該データをデータ利用部７０９に送る。データ利用部７０９は、例えば、当該マスター鍵を用いて当該データを復号し、復号結果を利用する。利用部７０９には、データを利用する為の適当なアルゴリズムが予め格納されているものとする。

【００２９】

【発明が解決しようとする課題】本発明は、利用デバイスにおけるデータの利用を、制御データによって制御する技術であって、従来技術に一般的に見られる二つの問題点を解決するものである。即ち、

1. セキュリティ上の問題

データ利用が利用デバイスにおける判断にのみ依存する方式は、リボケーションの実効性そのものに疑義がある。

2. リボケーションの錯誤の問題

リボケーションの錯誤は一種の「冤罪」であり、一般の善良な利用者の便宜を著しく損ねる。利用デバイス供給・販売業者にとっては、製品を巡るトラブルの要因になる可能性があり、看過できない。本発明は、上記二つの問題点を同時に解消することを目的とする。

【００３０】

【課題を解決するための手段】上記課題を解決するために、情報処理装置に割り当てられる複数の数の組からなるデバイス識別子を入力し、このデバイス識別子の部分経路に対応する経路関数値を、少なくとも一つ生成するようにした。

【００３１】また、制御データを生成する制御データ生成方法であって、入力された経路に属する数字の一部または全部を用いて作られる部分経路を、少なくとも一つ出力するようにした。

【００３２】また、データ利用を行う情報利用装置に、デバイス識別子と、デバイス識別子の一部または全部の数字を並べて作られる部分経路に対応する数値の少なくとも一つを格納するようにした。

【００３３】これらによって、利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避する事が可能となった。

【００３４】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【００３５】本発明の詳細な説明の前に、まずは本発明の概要を具体例を挙げて直感的に説明する。

【００３６】各利用デバイスは、4個の数字からなるIDを与えられているものとする。各数字は0、1または2の値をとる。従って、今の場合、IDの総数は34 =81個である。リボーク主体は、3行4列の行列Kを用意する。Kの成分は、例えば、乱数発生器などで生成した非負整数の乱数とする。Kのi行j列成分をk(i, j)と表記する。Kをデバイス・キー行列と呼び、Kの成分をデバイス・キー（デバイス鍵）と呼ぶ。

【００３７】リボーク主体は、各デバイスに対して、デバイス鍵の列を次のように割り当てる：

(ρ(1), ρ(2), ρ(3), ρ(4))

と言うデバイスのIDを持つデバイスに対して、デバイス鍵の列

[k(ρ(1), 1), k(ρ(2), 2), k(ρ(3), 3), k(ρ(4), 4)]

を割り当てる。

【００３８】例えば、(0, 2, 0, 1)と言うデバイスIDを割り当てたデバイスに対して、次のデバイス鍵の列を割り当てる：

[k(0, 1), k(2, 2), k(0, 3), k(1, 4)].

各デバイスは、IDと共に、割り当てられたデバイス鍵の列を保持する。

【００３９】さて、リボーク対象デバイスが存在しない初期状態で、リボーク主体は次のような制御データを配布する：

[(0, Enc(k(0, 1), K)), (1, Enc(k(1, 1), K)), (2, Enc(k(2, 1), K))].…式１。

【００４０】ここに、Enc()は適当なアルゴリズムによる暗号化を示す関数である：Enc(w,X)は、鍵wによって、平文データXを暗号化した結果を表す。ここでは、暗号化鍵、平文及び暗号文は非負整数と見なす。なお、利用デバイスは、Enc()に対応する復号関数Dec()を具備しているものとする。 Dec(w, Enc(w, X)) = X、が任意の鍵wと任意のデータXに対して成立する。

【００４１】Kはマスター鍵である。Kは利用デバイスが

[((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)), ((1, 2, 1), a(1, 2, 1)), ((1, 2, 2), a(1, 2, 2)), ((1, 2, 0, 0), a(1, 2, 0, 0)), ((1, 2, 0, 2), a(1, 2, 0, 2))]･･･式２

ここで、a(x1) = Enc(PF(x1), K), a(x1, x2) = Enc(PF(x1, x2), K), a(x1, x2, x3) = Enc(PF(x1, x2, x3), K), a(x1, x2, x3, x4) = Enc(PF(x1, x2, x3, x4), K)とする。

【００４３】

ここで、PF(x1) = k(x1, 1), PF(x1, x2) = k(x1, 1) ◎ k(x2, 2), PF(x1, x2, x3) = k (x1, 1) ◎ k(x2, 2) ◎ k(x3, 3), PF(x1, x2, x3, x4) = k (x1, 1) ◎ k(x2, 2) ◎ k(x3, 3) ◎ k(x4, 4).… 式 3

ただし、非負整数x, yに対してx ◎ yはxとyを二進表現した場合の、排他的論理和を表すものとする。上記PFを経路関数と呼ぶ。

【００４４】上記制御データの作り方は、次の通りである。(1, 2, 0, 1)と言う数字の並びの先頭から、数字を1個、2個、3個、4個取ったものを並べる：(1), (1, 2), (1, 2, 0), (1, 2, 0, 1). この並びを、デ

(0), (2), (1, 0), (1, 1), (1, 2, 1), (1, 2, 2), (1, 2, 0, 0), (1, 2, 0, 2).･･･式 4

この数の並びの各要素を変数とする関数aの値を並べたものが、上記式 2である。上のデータの作り方は、木構造を考えると理解し易い。

【００４６】図１において、リボーク対象のデバイスID (1, 2, 0, 1)は点線の経路に対応する。また、式 4の経路を実線で示す。それらは点線を辿って、最後だけ実線を辿る。図１の点線と実線と、その始／終点は、所謂三本木をなしている。この木をデバイスID (1, 2, 0, 1)のリボケーション木と呼ぶ。

【００４７】ここで、重要なのは、4つの列全てを通る経路は、点線の経路を除いて、必ずどこかで実線を通らなければならない、と言う点である。式 4の経路の集合を、デバイスID (1, 2, 0, 1)の境界集合と呼ぶ。デバイスID (1, 2, 0, 1)のリボケーション木の頂点集合は、当該デバイスIDに随伴する頂点集合と、当該デバイスIDの境界集合の和集合である。式 2の制御データは、デバイスID (1, 2, 0, 1)の境界集合に属する各経路に対して、非負整数値を対応させるものである。これをデータを利用する為に必要な情報である。式 1の制御データを初期制御データと呼ぶ。初期制御データは、三つの非負整数からなる。

【００４２】今、(1, 2, 0, 1)というデバイスIDを持つデバイスをリボークする事を考える。このとき、リボーク主体は、次の制御データを作り、利用デバイスに配布する：

[((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)), ((1, 2, 1), a(1, 2, 1)), ((1, 2, 2), a(1, 2, 2)), ((1, 2, 0, 0), a(1, 2, 0, 0)), ((1, 2, 0, 2), a(1, 2, 0, 2))]･･･式２

ここで、a(x1) = Enc(PF(x1), K), a(x1, x2) = Enc(PF(x1, x2), K), a(x1, x2, x3) = Enc(PF(x1, x2, x3), K), a(x1, x2, x3, x4) = Enc(PF(x1, x2, x3, x4), K)とする。

【００４３】

ここで、PF(x1) = k(x1, 1), PF(x1, x2) = k(x1, 1) ◎ k(x2, 2), PF(x1, x2, x3) = k (x1, 1) ◎ k(x2, 2) ◎ k(x3, 3), PF(x1, x2, x3, x4) = k (x1, 1) ◎ k(x2, 2) ◎ k(x3, 3) ◎ k(x4, 4).… 式 3

ただし、非負整数x, yに対してx ◎ yはxとyを二進表現した場合の、排他的論理和を表すものとする。上記PFを経路関数と呼ぶ。

【００４４】上記制御データの作り方は、次の通りである。(1, 2, 0, 1)と言う数字の並びの先頭から、数字を1個、2個、3個、4個取ったものを並べる：(1), (1, 2), (1, 2, 0), (1, 2, 0, 1). この並びを、デ

(0), (2), (1, 0), (1, 1), (1, 2, 1), (1, 2, 2), (1, 2, 0, 0), (1, 2, 0, 2).･･･式 4

この数の並びの各要素を変数とする関数aの値を並べたものが、上記式 2である。上のデータの作り方は、木構造を考えると理解し易い。

【００４６】図１において、リボーク対象のデバイスID (1, 2, 0, 1)は点線の経路に対応する。また、式 4の経路を実線で示す。それらは点線を辿って、最後だけ実線を辿る。図１の点線と実線と、その始／終点は、所謂三本木をなしている。この木をデバイスID (1, 2, 0, 1)のリボケーション木と呼ぶ。

【００４７】ここで、重要なのは、4つの列全てを通る経路は、点線の経路を除いて、必ずどこかで実線を通らなければならない、と言う点である。式 4の経路の集合を、デバイスID (1, 2, 0, 1)の境界集合と呼ぶ。デバイスID (1, 2, 0, 1)のリボケーション木の頂点集合は、当該デバイスIDに随伴する頂点集合と、当該デバイスIDの境界集合の和集合である。式 2の制御データは、デバイスID (1, 2, 0, 1)の境界集合に属する各経路に対して、非負整数値を対応させるものである。この対応をリボケーション関数と呼ぶ。この例では、aがリボケーション関数である。

【００４８】式 2の制御データを読取った利用デバイスの処理を述べる。利用デバイスのデバイスIDが(1, 2, 1, 1)であったとする。まず、利用デバイスは、デバイスIDの先頭から一つの数をとって得られる経路(1)が制御データに含まれるか否か調べる。即ち、経路(1)が、デバイスID (1, 2, 0, 1)の境界集合に属するか否かを調べる。当該制御データは、経路(0)及び(2)（とそれぞれの経路に対するリボケーション関数値）を含むが、(1)を含まない。この場合、利用デバイスは、デバイスIDの先頭から二つの数をとって得られる経路(1, 2)について、この経路が制御データに含まれるか否かを調べる。この様に、利用デバイスは、自分自身のデバイスIDに沿って、図１のリボケーション木を順に辿って行く。

【００４９】利用デバイスが、(1, 2)から(1, 2, 1)まで辿るときに、初めてリボケーション木の実線の枝

を通過する。即ち、経路(1, 2, 1)はデバイスID (1, 2,0, 1)の境界集合に属しており、従って、当該経路と当該経路に対応するリボケーション関数値が制御データに含まれている。利用デバイスは、当該リボケーション関数値a(1, 2, 1)を読取り、次の値を計算する。

Dec(PF(1, 2, 1), a(1, 2, 1)).･･･式 5

利用デバイスは、予め割り当てられたデバイス鍵の列[k(1, 1), k(2, 2), k(1, 3), k(1, 4)]を保持しているので、PF(1, 2, 1)を式 3に従って計算する事ができる。従って、利用デバイスは、確かに式 5の値を計算する事ができる。式 5の値は、 Dec(PF(1, 2, 1), Enc(PF(1, 2, 1), K)) = K.に等しい。

【００５０】従って、当該利用デバイスはマスター鍵を得て、データを利用する事ができる。即ち、リボークされていない。

【００５１】一方、利用デバイスがデバイスID (1, 2, 0, 1)を持っていたとする。上記と同じ手続きにより、利用デバイスは、自分自身のデバイスIDに沿って、図１のリボケーション木を順に辿って行く。今の場合は、しかし、リボケーション木の実線を通る事はない。経路(1), (1, 2), (1, 2, 0), (1, 2, 0, 1)は全て点線しか通らない。従って、どの経路も制御データ中に、対応するリボケーション関数値を見出すことができない。このデバイスは結局マスター鍵Kを得る事無く処理を終える。即ち、このデバイスはリボークされる。

【００５２】次に、デバイスID (1, 2, 0, 1)に加えて、デバイスID (1, 1, 1, 0)を持つデバイスをリボークする場合を考える。(1, 2, 0, 1)に随伴する頂点集合と(1, 1, 1, 0)に随伴する頂点集合の和集合を、(1, 2, 0, 1)及び(1, 1, 1,0)に随伴する頂点集合と言う。

【００５３】また、(1, 2, 0, 1)のリボケーション木の頂点集合と(1, 1, 1, 0)のリボケーション木の頂点集合の和集合を、(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合とする。(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合から、(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合を除いたものを、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合として定義する。

【００５４】具体的には、(1, 1, 1, 0)に随伴する頂点集合は、(1), (1, 1), (1,1, 1), (1, 1, 1, 0)であるから、(1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合は、次のようになる。

(1), (1, 1), (1, 2), (1, 1, 1), (1, 2, 0), (1, 1, 1, 0), (1, 2, 0, 1).

また、(1, 1, 1, 0)の境界集合は、(0), (2), (1, 0), (1, 2), (1, 1, 0), (1, 1, 2), (1, 1, 1, 1), (1, 1, 1, 2)であるから、(1, 2, 0, 1)及び(1, 1, 1, 0)のリボケーション木の頂点集合は、下記のようになる。

(0), (1), (2), (1, 0), (1, 1), (1, 2), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, 0), (1, 2, 1), (1, 2, 2),(1, 1, 1, 0), (1, 1, 1, 1), (1, 1, 1, 2),(1, 2, 0, 0), (1, 2, 0, 1),. (1, 2, 0,2).

従って、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合は、下記の通りである。

(0), (2), (1, 0), (1, 1), (1, 1, 0), (1, 1, 2), (1, 2, 1),(1, 2, 2),(1, 1, 1, 1), (1, 1, 1, 2), (1, 2, 0, 0), (1, 2, 0,2).

図２に二つのデバイスID (1, 2, 0, 1)及び(1, 1, 1, 0)に随伴する頂点集合と境界集合とを図示する。頂点集合は点線の辺を通る経路の全体に一致する。また、境界集合は点線の辺を辿って最後が実線を通る経路の全体に一致する。

【００５５】利用デバイスは、リボーク対象のデバイスIDが一つの場合と同じ手続きで、自分自身のデバイスIDに沿って、図１のリボケーション木を順に辿って行く。

【００５６】利用デバイスのデバイスIDが(1, 2, 1, 1)であったとする。利用デバイスの処理が経路(1, 2)から経路(1, 2, 1)に移動する際に、実線を通過する。即ち、境界集合に含まれる経路の中で、利用デバイスが始めて出会うものが(1, 2, 1)である。先ほどと全く同様に、制御データに含まれる、対応するリボケーション関数値に対する復号処理により、利用デバイスは、マスター鍵Kを得る事ができる。即ち、当該利用デバイスはリボークされていない。

【００５７】一方、利用デバイスのデバイスIDが(1, 1, 1, 0)であったとする。この利用デバイスによる制御データの処理は、デバイスID (1, 1, 1, 0)に随伴する頂点集合だけを通る。それは決して、(1, 2, 0, 1)及び(1, 1, 1, 0)の境界集合に含まれる経路を通らない。当該利用デバイスはリボケーション関数の値を得る事ができず、従って、マスター鍵を得ることはできない。即ち、当該利用デバイスはリボークされる。

【００５８】リボーク対象デバイスが三つ以上の場合についても、全く同様である。デバイスIDのリボケーション木を利用する方法によれば、錯誤の副作用を惹き起す事無く、リボーク対象デバイスを個別にリボークする事ができる。しかも、制御データの大きさ（要素数）は、（リボーク対象デバイスの数）×（デバイスIDの長さ）×2 を超えない。

【００５９】なお、デバイスIDをグループでリボークしたい場合も考えられる。例えば、或る利用デバイス・メーカーに(1, 2, *, *)と言うデバイスIDを割り当てたとする。ここに*はワイルド・カード（数字0, 1または2のいずれか）である。この場合、(1, 2, *, *)に随伴する頂点集合は、次の様になる。

(1), (1, 2), (1, 2, *), (1, 2, *, *)

(1, 2, *, *)の境界集合は、下記の様になる。

(0), (2), (1, 0), (1, 1).
図３に、(1, 2, *, *)に随伴する頂点集合と境界集合とを図示する。
【００６０】制御データは、従って、次の様に与えられる。
((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)).
この制御データによって、デバイスID (1, 2, *, *) を持つデバイスが、そして、それらだけがリボークされる。リボークの対象となるデバイスのグループが二つ以上ある場合も、制御データを個別デバイスのリボークと同様に定義する事により、錯誤の無いリボケーションが実現される。

【００６１】なお、以上では、制御データは、リボーク対象経路（群）が定める境界集合と、当該集合上のリボケーション関数値からなる例を考えた。リボケーション関数の定義域をリボーク対象経路（群）に随伴する頂点集合に拡張し、制御データを、リボケーション木の頂点集合と、当該頂点集合上のリボケーション関数値の組として定義する事もできる。この場合、上述の例より効率は落ちるが、ほぼ同様の効果が得られる。上述の例に則して、これを述べる。
【００６２】先の例と同様、(1, 2, 0, 1)というデバイスIDを持つデバイスをリボークする事を考える。このとき、リボーク主体は、次の制御データを作り、利用デバイスに配布する。

[((0), a(0)), ((2), a(2)), ((1, 0), a(1, 0)), ((1, 1), a(1, 1)),((1, 2, 1), a(1, 2, 1)), ((1, 2, 2), a(1, 2, 2)),((1, 2, 0, 0), a(1, 2, 0, 0)), ((1, 2, 0, 2), a(1, 2, 0, 2)),((1), b(1)), ((1, 2), b(1, 2)),((1, 2, 0), b(1, 2, 0)), ((1, 2, 0, 1), b(1, 2, 0, 1))]
･･･式６

ここで、
b(x1) = Enc(PF(x1), null),
b(x1, x2) = Enc(PF(x1, x2), null),
b(x1, x2, x3) = Enc(PF(x1, x2, x3), null),
b(x1, x2, x3, x4) = Enc(PF(x1, x2, x3, x4), null)
とする。nullは、マスター鍵が得られない事を示す、予め定められている数値である。今の場合、図１で点線の経路にもリボケーション関数値が割り当てられており、その値を定める関数がbである。
【００６３】利用デバイスの処理は、先の例と殆ど同様であるが、制御データのチェック方法だけが異なっている。利用デバイスのデバイスIDが(1, 2, 1, 1)であったとする。まず、利用デバイスは、デバイスIDの先頭から一つの数をとって得られる経路(1)に対するリボケーション関数値b(1)を制御データから読み取り、次の値を計算する。
Dec(PF(1), b(1)).
この値は、下記に等しい。
Dec(PF(1), Enc(k1(1), null)) = null.
従って、この場合、利用デバイスは、デバイスIDの先頭から二つの数をとって得られる経路(1, 2)について、同じ処理を繰り返す。利用デバイスが、(1, 2, 1)まで辿ったときに、初めてリボケーション木の実線を通過する。この場合の処理も同様である。ただし、今回はデバイス鍵Kを得る。
【００６４】一方、利用デバイスがデバイスID (1, 2, 0, 1)を持っていたとする。上記と同じ手続きにより、利用デバイスは、自分自身のデバイスIDに沿って、図１のリボケーション木を順に辿って行く。今の場合は、しかし、リボケーション木の実線を通る事はない。経路(1), (1, 2), (1, 2, 0), (1, 2, 0, 1)は全て点線しか通らない。従って、どの経路についても、リボケーション関数値の復号によって、nullを得るだけである。このデバイスは結局マスター鍵Kを得る事無く処理を終える。即ち、このデバイスはリボークされる。
【００６５】以下に、本発明の第１の実施の形態を詳細に説明する。
【００６６】図４は、デバイス情報生成装置１０を示したものである。
【００６７】リボーク主体は、デバイス情報生成装置１０を用いて、デバイスにデバイスIDとデバイス鍵列とを割り当てる。各デバイスにデバイス鍵列を与えることによって、各デバイスは経路関数k1, k2, k3を計算することが可能である。本実施の形態のデバイス情報生成装置１０は、デバイス鍵列ではなく、各デバイス経路関数値を直接与えるようにしたものである。こうすると、各デバイスはリボーク処理の際に、経路関数値計算の手間を省く事ができる。
【００６８】デバイス情報生成装置１０の動作の一例を図６、図７及び図８を用いて説明する。なお、デバイスIDは、各数字が0、1または2の値を取りうる3個の数字からなる長さ4であるとする。
【００６９】まず、入力部１０１がデバイス情報生成要求を受け取り、当該要求をＣＰＵ１０２へ送る（Ｓ１２０１）。ＣＰＵ１０２が乱数発生部１０４に、乱数発生を指示すると（Ｓ１２０２）、乱数発生部１０４により R1, R2, R3)を発生し、それらをＣＰＵ１０２へ送る（Ｓ１２０３）。ＣＰＵ１０２は、送られた R1, R2, R3 を作業メモリ１０３に格納する（Ｓ１２０４）。
【００７０】次に、ＣＰＵは数の組(R1, R2, R3)を既出力ＩＤ格納部１１０から検索する（Ｓ１２０５）。検

索により、(R1, R2, R3)が見つかった否かを判定し（Ｓ１２０６）、見つかった場合には、ステップＳ１２０３に戻り、再度乱数の発生を行う。一方、(R1, R2, R3)が見つから無かった場合には、(R1, R2, R3)を既出力ＩＤ格納部１１０に格納する（Ｓ１２０７）。また、(R1, R2, R3)をデバイスＩＤ格納部１０９へも格納する（Ｓ１２０８）。

【００７１】ＣＰＵ１０２は、R1を経路関数計算部１０６に送り（Ｓ１２０９）、経路関数計算部１０６にてPF(R1)値を計算する（Ｓ１２１０）。ＣＰＵ１０２は、計算されたPF(R1)値をデバイスＩＤ格納部１０９へ格納する（Ｓ１２１１）。

【００７２】次に、ＣＰＵ１０２は、(R1, R2)を経路関数計算部１０６に送り（Ｓ１２１２）、経路関数計算部１０６にてPF(R1, R2)値を計算する（Ｓ１２１３）。ＣＰＵ１０２は、計算されたPF(R1, R2)値をデバイスＩＤ格納部１０９へ格納する（Ｓ１２１４）。

【００７３】同様に、ＣＰＵ１０２は、(R1, R2, R3)を経路関数計算部１０６に送り（Ｓ１２１５）、経路関数計算部１０６にてPF(R1, R2, R3)値を計算する（Ｓ１２１６）。ＣＰＵ１０２は、計算されたPF(R1, R2, R3)値をデバイスＩＤ格納部１０９へ格納する（Ｓ１２１７）。

【００７４】次に、ＣＰＵ１０２は、デバイスＩＤ格納部１０９からデバイスＩＤ(R1, R2, R3)を読み出して、出力部１０５へ送る（Ｓ１２１８）。また、デバイスＩＤ格納部１０９から経路関数値PF(R1)、PF(R1, R2)、PF(R1, R2, R3)を読み出し、出力部１０５へ送る（Ｓ１２１９）。これにより、出力部１０５は、デバイスＩＤ(R1, R2, R3)及び経路関数値PF(R1)、PF(R1, R2)、PF(R1, R2, R3)を出力する（Ｓ１２２０）。

【００７５】経路(R1)、(R1, R2)及び(R1, R2, R3)をデバイスID (R1, R2, R3)の部分経路と呼ぶことがある。本実施の形態では、各部分経路に対する経路関数値は次のようにとられる。

PF(R1) = Enc(k(R1, 1), 1)

PF(R1, R2) = Enc(k(R2, 2), k1(R1)),

PF(R1, R2, R3) = Enc(k(R3, 3), k2(R1, R2))

ここで、

k(0, 1) k(0, 2) k(0, 3)

k(1, 1) k(1, 2) k(1, 3)

k(2, 1) k(2, 1) k(2, 3)

は、デバイス・キー行列であり、予め乱数発生器などによって生成され、デバイス・キー行列格納部１１１に格納されている。

【００７６】上記したデバイス制御生成装置１０の動作の一例において、経路関数値を計算すべき経路pは、(p(1))、(p(1), p(2))または、(p(1), p(2), p(3))と言う形で、経路関数計算部１０６に入力される。なお、各p(j) (j= 1, 2, 3)は、数字0, 1または2である。経路関数計算部１０６は、入力された経路の長さに応じて、PF (p(1))、PF(p(1), p(2))またはPF(p(1),p(2), p(3))の値を計算し、出力する。この経路関数計算部１０６の動作の一例を図５を用いて説明する。

【００７７】まず、経路ｐを受け取る（Ｓ１１０１）。ｐの長さを変数１に、Ｖ＝１、Ｊ＝１にそれぞれ初期設定する（Ｓ１１０２～Ｓ１１０４）。次にＪが１より大きいか否かを判定する（Ｓ１１０５）。この場合、Ｊ＝１なので、否と判定され、数の組(p(j), j)を鍵読み取り部１０８に送る（Ｓ１１０６）。次に、k(p(j), j)を鍵読み取り部１０８から受け取り（Ｓ１１０７）、これを鍵としてＶを暗号化し、その結果をＶに代入する（Ｓ１１０８）。次にＪを一だけ増加させ（Ｓ１１０９）、ステップＳ１１０５の判定に戻る。

【００７８】もし、ステップＳ１１０５によって、Ｊが１より大きいと判定されるとＶの値を出力し、（Ｓ１１１０）経路関数計算部１０６の動作は終了する。

【００７９】デバイス情報生成装置１０は、デバイス情報生成要求を受け取り、デバイスID：(R1, R2, R3)、経路関数値：(PF1, PF2, PF3)を出力する。これらは、それぞれ三つの符号無し整数の組である。経路関数値は、その作り方から明らかな様に、デバイスIDに依存して定まる。

PF1 = PF(R1)、

PF2 = PF(R1, R2)、

PF3 = PF(R1, R2, R3)。

【００８０】本実施形態のデバイス情報生成装置１０は、ＭＫＢのデバイス情報生成装置１０と対比されるべきものである。ＭＫＢには、経路関数という概念が存在しない。即ち、ＭＫＢでは、デバイスIDが通過するＭＫＢの成分だけで、当該デバイスIDがリボークされるか否かが決まる。一方、本発明は経路と見なしたデバイスIDをリボークの対象とするので、デバイスIDが通過する数字を同定する為に、経路関数の値を利用する。本実施形態のデバイス情報生成装置１０は、以下の二点に特徴がある。

Ａ）経路関数計算部１０６を具備している。経路関数計算部１０６は、数字の並びであるデバイスIDについて、デバイスIDを構成する数字の一部または全部を用いて作られる部分経路に対する経路関数値を計算する。経路関数値は、経路の長さが1の場合を除き、経路に対応するデバイス・キー行列の複数の成分に依存して定められる数値である。

Ｂ）デバイスIDと共に、当該デバイスIDの部分経路に対応する経路関数値を出力する。

【００８１】次に、制御データ生成装置２０について説明する。図９は制御データ生成装置２０の構成を示した図である。

【００８２】デバイス・キー行列格納部２１６には、デバイス・キー行列が格納されている。これは、デバイス情報生成装置１０で使用されるデバイス・キー行列と同一のものである。

【００８３】まず、リボーク対象デバイスが存在しない初期状態における、制御データ生成装置２０の動作の一例を図１０に示す。

【００８４】始めに、マスター鍵ｋをマスター鍵入力部２０１により入力する（Ｓ１３０１）。ＣＰＵ２０５は、入力されたＫをマスター鍵格納部２０９に送る（Ｓ１３０２）。そして、Ｊを０に初期設定する（Ｓ１３０３）。

【００８５】次にＪが３より小さいか否かを判定し（Ｓ１３０４）、Ｊが３以上になるまで以下の処理を繰り返す。

【００８６】まず、ＣＰＵ２０５は、マスター鍵格納部２０９からマスター鍵Ｋを読む（Ｓ１３０５）。また、ＣＰＵ２０５は、数の組（Ｊ，1）を制御データ格納部２１１に格納する（Ｓ１３０６）。この数の組（Ｊ，1）を鍵読み取り部２１５に送る（Ｓ１３０７）。

【００８７】鍵読み取り部２１５は、デバイス・キー行列格納部２１６からk（Ｊ，1）を読み、ＣＰＵ２０５へ送る（Ｓ１３０８）。ＣＰＵ２０５はk（Ｊ，1）と、Ｋとを暗号化部２０７に送る（Ｓ１３０９）。

【００８８】暗号化部２０７は、Ｋをk（Ｊ，1）で暗号化し、得られた結果a（Ｊ，1）をＣＰＵ２０５へ送る（Ｓ１３１０）。

【００８９】ＣＰＵ２０５は、経路と数値の組（（Ｊ），1 a（Ｊ，1））を出力部２０４へ送り（Ｓ１３１１）、出力部２０４は、これを出力する（Ｓ１３１２）。

【００９０】Ｊを１だけ増加させ（Ｓ１３１３）、ステップＳ１３０４へ戻る。ステップＳ１３０４では、Ｊが３以上になると上記の繰り返しを止め、終了する。

【００９１】初期状態における、制御データ生成装置２０の出力は、（（０），a（０）），（（１），a（１）），（（２），a（２））である。ここに、（０）、（１）、（２）はそれぞれ唯一つの数字からなる（長さ1の）経路である。また、a（Ｊ）は、Enc(k(Ｊ，１)，Ｋ)を示している。

【００９２】リボーク対象経路は、数の並びの形でリボーク情報入力部２０２に入力される。（2，0，１）あるいは（1，１）などがリボーク対象経路pとして、リボーク情報入力部２０２に入力される。

【００９３】次に、リボーク対象経路pが初めて入力された時の動作を図１１、図１２及び図１３に示す。

【００９４】まず、リボーク情報入力部２０２に経路ｐが入力され（Ｓ１４０１）、作業メモリ２０６に書き込まれる（Ｓ１４０２）。ＣＰＵ２０５は、ｐを境界集合計算部２０８へ送り（Ｓ１４０３）、境界集合計算部２０８がｐの境界集合Ｖ’を求める（Ｓ１４０４）。得られた境界集合Ｖ’を受けたＣＰＵ２０５は、Ｖ’を境界集合格納部２１０に格納する（Ｓ１４０６）。

【００９５】次に、ＣＰＵ２０５は、Ｖ’の要素数を変数Ｎに設定し（Ｓ１４０７）、Ｊを１に設定する（Ｓ１４０８）。

【００９６】これ以降、ＪがＮより大きいか否かを判定し（Ｓ１４０９）、この判定結果が否であれば、ステップＳ１４１０～Ｓ１４１９を実行する。すなわち、

・ＣＰＵ２０５が、V'のJ番目の経路p[Ｊ]を経路関数計算部２１３に送る（Ｓ１４１０）

・経路関数計算部２１３がp[Ｊ]における経路関数値k（p[Ｊ]）を計算する（Ｓ１４１１）

・ＣＰＵ２０５が、k（p[Ｊ]）を経路関数計算部２１３から受け取る（Ｓ１４１２）

・ＣＰＵ２０５が、マスター鍵Ｋをマスター鍵格納部２０９から読み取る（Ｓ１４１３）

・ＣＰＵ２０５が、数値の組（k（p[Ｊ]，Ｋ）を暗号化部２０７に送る（Ｓ１４１４）

・暗号化部２０７がk（p[Ｊ]）を鍵としてKを暗号化。その結果a（p[Ｊ]）をＣＰＵ２０５に返す（Ｓ１４１５）

・ＣＰＵ２０５が経路と数値の組（p[Ｊ]，a（p[Ｊ]））を制御データ格納部２１１に追加（Ｓ１４１６）

・ＣＰＵ２０５が（p[Ｊ]，a（p[Ｊ]））を出力部２０４に送る（Ｓ１４１７）

・出力部２０４が（p[Ｊ]，a（p[Ｊ]））を出力（Ｓ１４１８）

・Ｊを１だけ増分する（Ｓ１４１９）

をＪがＮより大きくなるまで繰り返す。

【００９７】ステップ１４０９において、ＪがＮより大きくなったと判断された場合、ＣＰＵ２０５がｐを作業メモリ２０６から読み出し（Ｓ１４２０）、随伴頂点集合計算部２１４に送る（Ｓ１４２１）。随伴頂点集合計算部２１４がｐの随伴頂点集合Ｖを求め（Ｓ１４２２）、これをＣＰＵ２０５が受ける（Ｓ１４２３）。この受け取った随伴頂点集合Ｖを随伴頂点集合格納部２１２に格納し（Ｓ１４２４）、処理を終える。

【００９８】以上の動作について解説を加える。リボーク対象経路pを初めて入力する時点で、制御データ生成装置２０の境界集合格納部２１０と随伴頂点集合格納部２１２には何も格納されていない。同様に、制御データ格納部２１１の内容も空である。先に述べた様に、初期状態についても制御データ自体は存在する。制御データ生成装置２０は、初期状態で、それを出力するが、当該制御データは制御データ格納部２１１に格納されない。

【００９９】リボーク対象経路pが入力されると、制御データ生成装置２０のＣＰＵ２０５は、pを境界集合計算部２０８に送る。境界集合計算部２０８は、pの境界集合に属する経路を全て求めて、ＣＰＵ２０５に返す。pの境界集合とは、pの先頭からとった部分経路の集合に

含まれる各経路の最後の数字を、異なった数字に置き換える事で得られる経路の集合である。例えば、p = (2, 0, 1)に対して、pの先頭からとった部分経路の集合は、(2), (2, 0), (2, 0, 1)となるから、境界集合は、(0), (1), (2, 1), (2, 2), (2, 0, 0), (2, 0, 2) の各経路の集合である。

【０１００】ＣＰＵ２０５は、境界集合計算部２０８から受け取った経路を境界集合格納部２１０に格納する。

【０１０１】ＣＰＵ２０５は、次いで、境界集合に属する各経路に対するリボケーション関数値を求める。リボケーション関数aは、経路(x1)、(x1, x2)、(x1, x2, x3)に対して、それぞれ次の様に定義される。

a(x1) = Enc(PF(x1), K),

a(x1, x2) = Enc(PF(x1, x2), K),

a(x1, x2, x3) = Enc(PF(x1, x2, x3), K).

式 7のaは、長さ1の経路に関するリボケーション関数である。ここで、Kはマスター鍵、PFは経路関数である。各経路におけるPFの値は経路関数計算部で計算される。本制御データ生成装置２０における経路関数計算部２１３は、図４のデバイス情報生成装置１０における経路関数計算部１０６と同一のものである。制御データ生成装置２０は、境界集合に属する各経路と、当該経路における経路関数値の組を全て出力する。これらの組の全体が、入力されたリボーク対象経路に対する制御データとなる。例えば、リボーク対象経路p = (2, 0, 1)に対する制御データ生成装置２０の出力は、次の様になる。

((0), a(0)), ((1), a(1)), ((2, 1), a(2, 1)), ((2, 2), a(2, 2)),

((2, 0, 0), a(2, 0, 0)), ((2, 0, 2), a(2, 0, 2)).

次に、ＣＰＵ２０５は、経路pを、随伴頂点集合計算部２１４に送る。随伴頂点集合計算部２１４は、pの随伴頂点集合に属する経路を全て求めて、CPUに返す。デバイスID pの随伴頂点集合とは、pの先頭から順にとった部分経路の集合である。即ち(p(1), p(2), p(3))の随伴頂点集合は、(p(1)), (p(1), p(2)), (p(1), p(2), p(3))の三つの経路の集合である。

【０１０２】長さがデバイスIDの長さ（本例では3）に満たない経路に関して、随伴頂点集合は次のように定義される。(p(1))の随伴頂点集合は、(p(1))及び(p(1), ?, ?)の形で書ける全ての経路からなる。(p(1), p(2))の随伴頂点集合は、(p(1))、(p(1), p(2))及び(p(1), p(2), ?)の形で書ける全ての経路からなる。なお、「?」は0、1または2の任意の数字を表すワイルド・カードである。

【０１０３】要するに、デバイスIDの長さに満たない経路の随伴頂点集合は、当該経路の部分経路と、デバイスIDの長さまでワイルド・カード?を補った形で書ける経路の全体である。デバイスID(2, 0, 1)に随伴する頂点集合は、(2), (2, 0), (2, 0, 1)の三つの経路からなる。

【０１０４】また、例えば経路(1, 1)の随伴頂点集合は、(1), (1, 1), (1, 1, 0), (1, 1, 1), (1, 1, 2)の5つの経路からなる。

【０１０５】ＣＰＵ２０５は随伴頂点集合計算部２１４から受け取った経路を、随伴頂点集合格納部２１２に格納する。

【０１０６】以上、初期状態の制御データ生成装置２０の動作と、初めてリボーク対象経路を入力した際の制御データ生成装置２０の動作を述べた。

【０１０７】次に、二つ以上の経路をリボークする場合について、図１４乃至図１８のフローチャートを用いて、動作の一例を示す。

【０１０８】まず、リボーク情報入力部に経路pを入力（Ｓ１５０１）し、ＣＰＵ２０５がpを読み取り、作業メモリに格納（Ｓ１５０２）する。次に、ＣＰＵ２０５がpを境界集合計算部２０８に送る（Ｓ１５０３）。境界集合計算部２０８は、pの境界集合V'を計算（Ｓ１５０４）する。ＣＰＵ２０５は、計算されたV'を受け取り作業メモリに格納（Ｓ１５０５）する。

【０１０９】ＣＰＵ２０５は、pを随伴頂点集合計算部２１４に送る（Ｓ１５０６）。随伴頂点集合計算部２１４は、pの随伴頂点集合Vを計算（Ｓ１５０７）する。ＣＰＵ２０５は、その計算されたVを受け取り、作業メモリに格納（Ｓ１５０８）する。ＣＰＵ２０５は、境界集合格納部から経路集合U'を読み出す（Ｓ１５０９）。ＣＰＵ２０５は、差集合U' － Vを求め作業メモリに格納（Ｓ１５１０）する。ＣＰＵ２０５は、制御データ格納部からU' － Vに含まれない経路と、対応するリボケーション関数値とを削除する（Ｓ１５１１）。

【０１１０】次に、ＣＰＵ２０５は、随伴頂点集合格納部から経路集合Uを読み出す（Ｓ１５１２）。また、ＣＰＵ２０５は作業メモリからV'を読み出す（Ｓ１５１３）。ＣＰＵ２０５は、差集合V' － Uを求め、作業メモリに格納（Ｓ１５１４）する。ＣＰＵ２０５が、V' － Uに含まれる経路の数を数え、経路数を変数Nに設定する（Ｓ１５１５）。その後、ここで、Ｊを１に初期設定（Ｓ１５１６）する。

【０１１１】次に、ＪがＮより大きいか否か判断し（Ｓ１５１７）、ＪがＮより大きくなるまで以下のステップＳ１５１８～Ｓ１５２８を繰り返す。

【０１１２】すなわち、

・ＣＰＵ２０５が、V' － UのJ番目の経路q[Ｊ]を経路関数計算部に送る（Ｓ１５１８）。

・経路関数計算部が経路関数値PF(q[Ｊ])を計算する（Ｓ１５１９）。

・ＣＰＵ２０５がPF(q[Ｊ])を受けとる（Ｓ１５２０）。

・ＣＰＵ２０５は、マスター鍵Kをマスター鍵格納部か

ら取得する（Ｓ１５２１）。
・ＣＰＵ２０５が、数の組(PF(q[Ｊ]), K)を暗号化部に送る（Ｓ１５２２）。
・暗号化部が、KをPF(q[Ｊ])で暗号化結果a(q[Ｊ])を得る（Ｓ１５２３）。
・ＣＰＵ２０５は、a(q[Ｊ])を暗号化部から取得する（Ｓ１５２４）。
・ＣＰＵ２０５が、経路と数の組(q[Ｊ], a(q[Ｊ]))が制御データ格納部に存在するか否か調べ（Ｓ１５２５）、存在すると判断した場合のみ、ＣＰＵ２０５が、(q[Ｊ], a(q[Ｊ]))を制御データ格納部に追加（Ｓ１５２７）する。
・Ｊを1増分する。（Ｓ１５２８）。
の処理をＪがＮより大きくなるまで繰り返し行なう。

【０１１３】ステップＳ１５１７において、Ｊが大きくなったら、次に、ＣＰＵ２０５が、作業メモリ上の経路集合V'の要素数を数え、その数を変数Nに設定する（Ｓ１５２９）。そして、Ｊを再度1に設定し直す（Ｓ１５３０）。

【０１１４】次に、Ｊがｎより大きいか否かを判断し（Ｓ１５３１）、大きくないならば、大きくなるまで、ＣＰＵ２０５が、V'のJ番目の経路q[Ｊ]が境界集合格納部に存在するか否か調べ（Ｓ１５３２）、存在しない場のみq[Ｊ]を境界集合格納部に追加（Ｓ１５３４）する。そして、ｊを１だけ増分し、ステップＳ１５３１に戻る。ステップＳ１５３１において、ＪがＮより大きくなると、上記のステップＳ１５２９からステップＳ１５３５までと同様の処理を経路集合Vに対しても行なう（Ｓ１５３６～Ｓ１５４２）。そして終了する。

【０１１５】以上が、二つ以上の経路をリボークする場合の動作の一例である。これを以下に解説する。

【０１１６】二つ目以降のリボーク対象経路pを入力する時点で、制御データ生成装置２０の境界集合格納部２１０と随伴頂点集合格納部２１２には、それぞれ現在の制御データを求める際に計算した境界集合（の和集合）と頂点集合とが格納されている。同様に、制御データ格納部２１１には、現在の制御データが格納されている。

【０１１７】制御データ生成装置２０は、先ずpの境界集合V'(ｐ)と、pの随伴頂点集合V(p)とを求める。これらは、それぞれ境界集合計算部２０８と随伴頂点計算部２１４に経路pを送る事により求められる。ＣＰＵ２０５は、V'(ｐ)及びV(ｐ)を作業メモリ２０６に格納する。次いで、制御データ生成装置２０は以下の処理を順に行う。

a. 境界集合格納部２１０から経路の集合を読み出し、V(ｐ)に含まれていない経路のみを作業メモリ２０６に書きこんで行く。境界集合格納部２１０に格納されている経路の集合をU'とすると、作業メモリ２０６には差集合U' － V(ｐ)に属する経路が書きこまれる事になる。

b. 制御データ格納部２１１を調べ、作業メモリ２０６上の経路集合U' － V(ｐ)に含まれていない経路と、当該経路におけるリボケーション関数値の組を消去する。

c. 随伴頂点集合格納部２１２から経路の集合Uを読み出す。作業メモリ２０６上の経路集合V'(ｐ)の経路を順に選び、当該経路がUに含まれていなければ、当該経路を作業メモリ２０６にコピーする。この処理により、作業メモリ２０６上に差集合V'(ｐ) － Uが得られる。

d. 作業メモリ２０６上の経路集合V'(ｐ) － Uに属する各経路に関して、当該経路におけるリボケーション関数値を求める。そして、V'(ｐ) － Uに属する各経路と、当該経路におけるリボケーション関数値の組を、制御データ格納部２１１に追加する。この際、重複する組は追加しない。

e. 境界集合格納部２１０の経路集合U'に、V'(ｐ)に含まれる経路を追加する。この際、重複する経路は追加しない。これにより、U'の内容はU' ∪ V'(ｐ)に変わる。

f. 随伴頂点集合格納部２１２の経路集合Uに、V(ｐ)に含まれる経路を追加する。この際、重複する経路は追加しない。これにより、Uの内容はU ∪ V(ｐ)に変わる。

【０１１８】上述の処理a～dを行う前の段階で、制御データ格納部２１１に格納されている経路の集合をCとおく。即ち、制御データ格納部２１１は、Cに含まれる各経路と、当該経路におけるリボケーション関数値の組を格納していたとする。a～dの処理を行う事により、制御データ格納部２１１の経路集合は、次の様に更新される。

$$C' = (C \cap (U' - V(p))) \cup (V'(p) - U) \cdots 式8$$

処理a～dの終了後、制御データ格納部は、C'に属する各経路と、当該経路におけるリボケーション関数値の組を格納している。処理e及びfは、次のリボーク対象経路の追加に備える為のデータ更新作業である。

【０１１９】P[１], ..., p[ｓ]をリボーク対象経路とする。p[１], ..., p[ｓ]に対して、順に上記処理a～dを繰り返す事によって得られる制御データを(Xs, a(Xs))とする。すなわち、当該制御データは、経路の集合Xsに属する各経路と、当該経路におけるリボケーション関数値の組の集合になっている。このとき、次式の成立が証明される。

$$Xs = (V'(p[1]) \cup \cdots \cup V'(p[s])) - (V(p[1]) \cup \cdots \cup V(p[s])) \cdots 式9。$$

【０１２０】各経路p[１], ..., p[ｓ]を、境界集合計算部２０８と随伴頂点集合計算部２１４にそれぞれ入

力する事により、経路集合V'(p[1]), ..., V'(p[s])とV(p[1]), ..., V(p[s])が得られるから、式 9の右辺を直接構成する動作も勿論可能であり、その様にしても良い。本実施の形態では、しかし、式 9の右辺を漸化的に構成する方法を採った。

【０１２１】具体例に、上記制御データの構成手順を適用してみる。デバイスID (2, 0, 1)と経路(1, 1)をリボークする。先ず、(2, 0, 1)をリボークする制御データは、次の通り。

((0), a(0)) ((2, 1), a(2, 1)) ((2, 0, 0), a(2, 0, 0))((1), a(1)) ((2, 2), a(2, 2)) ((2, 0, 2), a(2, 0, 2))

この段階で、境界集合格納部２１０に格納されている経路集合U'は次の通りである。

(0) (2, 1) (2, 0, 0) (1) (2, 2) (2, 0, 2)

また、随伴頂点集合格納部２１２は、次の経路集合Uを格納している。

(2) (2, 0) (2, 0, 1)

さて、経路(1, 1)を境界集合計算部２０８に入力する事により、下記の境界集合V'(1, 1)を得る。

(0)(1, 0) (1, 2)

また、経路(1, 1)を随伴頂点集合計算部に入力する事により、下記の随伴頂点集合V(1, 1)を得る。

(1) (1, 1) (1, 1, 0) (1, 1, 1) (1, 1, 2)

上記処理aで作業メモリに書きこまれる経路集合U' - V(1, 1)は、従って、次の様になる。

(0) (2, 1) (2, 0, 0) (2, 2) (2, 0, 2)

処理bにより、制御データ格納部の内容は次のように更新される。

((0), a(0)) ((2, 1), a(2, 1)) ((2, 0, 0), a(2, 0, 0))((2, 2), a(2, 2)) ((2, 0, 2), a(2, 0, 2))

処理cの結果作業メモリに格納される経路集合V'(1, 1) - Uは次の様になる。

(0) (1, 0) (1, 2)

処理dの結果、制御データ格納部の内容、即ち制御データは次のように変化する。

((0), a(0)) ((2, 1), a(2, 1)) ((2, 0, 0), a(2, 0, 0))((2, 2), a(2, 2)) ((2, 0, 2), a(2, 0, 2)) ((1, 0), a(1,0)) ((1, 2), a(1, 2))

このデータが、制御データとして出力される。

【０１２２】更に、処理eにより、境界集合格納部の内容は次の様に更新される。

(0) (2, 1) (2, 0, 0) (1) (2, 2) (2, 0, 2) (0) (1, 0) (1, 2)

最後に処理fにより、随伴頂点集合格納部の内容は次の様に変化する。

(2) (2, 0) (2, 0, 1) (1) (1, 1) (1, 1, 0) (1, 1, 1) (1, 1, 2)

なお、処理dの結果更新された制御データ格納部が格納する経路集合は、確かに下記経路集合に一致する。

(V'(2, 0, 1) ∪ V'(1, 1)) - (V(2, 0, 1) ∪ V'(1, 1))

次に、随伴頂点集合計算部２１４の動作の一例を図１９及び図２０に示し、説明する。

【０１２３】初めに、随伴頂点集合計算部２１４は、経路pを受け取り（Ｓ１６０１）、経路pの長さを変数Lに格納する（Ｓ１６０２）。

【０１２４】次に、デバイスID長格納部からデバイスID長を読み取り、変数Nに格納する（Ｓ１６０３）。

【０１２５】そして、Ｎ－Ｌが０より大きいか否かの判定を行なう（Ｓ１６０４）。Ｎ－Ｌが０より大きければ、Ｊを１に設定する（Ｓ１６０５）。その後、ＪがＬより大きくなるまで、Ｊを１づつ、増分しながら経路(p(１), , p(Ｊ))を出力する（Ｓ１６０６～Ｓ１６０８）。

【０１２６】一方、ステップＳ１６０４にて、 Ｎ－Ｌが０より大きくなければ、以下の処理に進む。まずＪを1に設定する（Ｓ１６０９）。その後、ＪがＮ－Ｌより大きくなるまで、Ｊを１づつ増分（Ｓ１６１８）しながら、以下の処理を行う。

・J個の成分を持つ整数配列X, Yを用意する（Ｓ１６１１）。

・Y[0] = 2Y[Ｊ - １] = 2とする（Ｓ１６１２）。

・xを０に設定する（Ｓ１６１３）。

・x = Yとなるまで、Xを1づつ増分しながらxの三進表現を配列Xに格納し、経路(p(１), , p(Ｌ), X[０], , X[Ｊ - １])を出力する（Ｓ１６１４～Ｓ１６１７）。

なお、ステップＳ１６１０において、ＪがＮ－Ｌより大きくなった際に処理を終了する。

【０１２７】上記で説明した動作について、以下に解説する。

【０１２８】随伴頂点集合計算部２１４には、図示しない予めデバイスIDの長さが格納されているデバイスID長格納部を持っている。随伴頂点集合計算部２１４は、格納されたデバイスIDの長さを取得し、読みこんだ経路の長さと比較する。読みこんだ経路の長さLがデバイスIDの長さNより小さい場合、デバイスIDの長さと経路の長さの差をとり、その差N - Lに一致する要素数を持つ整数配列X, Yを用意する。

【０１２９】更に、Yの全ての成分を2に設定する。Yは終了条件判定用である。整数の変数xを0から3N-L - 1まで変化させながら、xの三進表現を配列Xに格納して行く。Xは0, ..., 0 (N - L個)から、2, ..., 2 (N - L個)まで変化する。

【０１３０】随伴頂点集合計算部２１４は、各xの値毎に、(p(１), ..., p(Ｌ), X[０], ..., X[Ｎ - Ｌ])を出力する。Xは、随伴頂点集合の要素の中で、ワイ

ルド・カードを用いて表現される要素を生成する。リボーク対象経路pの長さがデバイスIDの長さより小さいのは、(p(１), ..., p(Ｌ), ?, ..., ?)と表現される一群のデバイスIDを一括してリボークする場合及び、その場合に限られる。この種の一括リボケーションは、特定の製造業者による特定分野の製品を全てリボークする場合等に発生する。

【０１３１】一度にリボークされるデバイスIDの数は、従って、高々数百～数千万のオーダーである。そして、生成すべき随伴頂点集合の個数は、リボーク対象デバイスIDの数の約二倍である。この数は大きいが、しかし、この程度の数の随伴頂点集合を生成する処理は、現行の計算機で現実的な時間内に終了させる事ができる。しかも、この処理は、リボーク対象デバイス群が新たに追加された場合に、リボーク主体が保有する制御データ生成装置２０で一度だけ実行すれば良い性質のものである。

【０１３２】次に、境界集合計算部２０８の動作を図２１に示し説明する。

【０１３３】境界集合計算部２０８は、経路ｐを受け取って（Ｓ１７０１）この経路ｐの差が差を変数Ｌに格納する（Ｓ１７０２）。Ｊを１に設定し（Ｓ１７０３）、ＪがＬより大きくなるまで、次の処理を繰り返し行う。

・Ｉに０を設定する（Ｓ１７０５）。

・Ｉが３より小さいか否かを判定し（Ｓ１７０６）、小さければ、ＩはＰ（Ｊ）と同じかを判定し（Ｓ１７０７）、同じでなければ、経路(p(１), ..., p(J-1), Ｉ)を出力し（Ｓ１７０８）、Ｉを１だけ増分して、再度、ステップＳ１７０６の判定処理を行う。

・ステップＳ１７０６の判定処理で小さくないと判断されるとＪを1増分する。なお、ＪがＬより大きくなる終了となる。

【０１３４】上記動作を換言すると、境界集合計算部２０８は、受け取った経路pについて、経路pの最後の数字p(Ｌ)を、p(Ｌ)とは異なった数字に置き換えた経路を作り、それを出力する。経路の長さがLで、経路を構成する数字の範囲が0からm -1であるとき、境界集合に含まれる経路の総数はmLである。

【０１３５】本発明の制御データ生成装置２０は、従来のＭＫＢ技術におけるＭＫＢ生成装置と対比すべきものである。本発明の制御データ生成装置２０は、随伴頂点集合計算部２１４を具備する点に特徴がある。また、本発明の制御データ生成装置２０は、境界集合計算部２０８を具備する点に特徴がある。また、本発明の制御データ生成装置２０は、経路関数計算部２１３を具備する点に特徴がある。随伴頂点集合計算部：数字の並びである経路について、入力された経路に属する数字の一部または全部を用いて作られる部分経路、または当該経路に少なくとも一つの数字を追加して作られる経路を、少なくとも一つ出力する。境界集合計算部：入力された経路に属する数字の一部または全部を用いて作られる部分経路の少なくとも一つについて、当該部分経路に属する数字の一部または全部を変更して作られる経路を出力する。

【０１３６】また、本実施例の制御データ生成装置２０は、以下のデータを出力する点に特徴がある。

ー数字の並びである経路と、当該経路に対応付けられた数値を、少なくとも一組出力する。

ー特に、上記経路は、境界集合計算部の出力経路である。

ー特に、上記経路に対応付けられた数値は、当該経路におけるリボケーション関数値である。

ーリボケーション関数値：数字の並びである経路に対して定められる数値で、当該経路における経路関数値とマスター鍵に依存する数値である。

【０１３７】次にメディア上の制御データについて説明する。

【０１３８】制御データ生成装置２０の出力である制御データは、ネットワークや放送を通じて利用デバイスに配布される。あるいは、メディアに記録されて配布される事もある。制御データがメディアに記録されて配布される場合、制御データは、例えば、経路と当該経路におけるリボケーション関数値のペアを列挙する形式で、メディアに記録される。この場合、経路を数字の列として、辞書式に順序づけて列挙するようにしても良い。

【０１３９】図２２に一例を挙げる。経路はPDに格納される。PDは8ビットのNLフィールドとPTフィールドを持つ。NLフィールドには、経路を構成する数字の数が記録されている。その数をｖとする。PTフィールドは2*ｖビットの長さを持つ。本実施例において、経路を構成する数字は0, 1または2であるから、それは2ビットで表現される。例えば、経路(2, 0, １)の場合、NLフィールドの値は3であり、PTフィールドはこの6ビット"100001"のビット列である。

【０１４０】BPは、経路と、当該経路におけるリボケーション関数値の組を格納する。BPはPDフィールドとVRフィールドから構成される。PDフィールドについては既に説明した。VRフィールドは128ビットの符号なし整数である。制御データのフォーマットはCDと書かれているものである。CDフォーマットは、NBPフィールドと0個以上のBPフィールドから構成される。NBPフィールドには、制御データを構成するBPフィールドの数が書かれている。その数をnとすると、制御データは、BP 1からBP nまでn個のBPフィールドを持つ。

【０１４１】本発明の制御データが記録されたメディアは、少なくとも下記の情報を含む点に特徴がある。

ー数字の並びである経路と、当該経路に対応付けられた数値が、少なくとも一組含まれる。当該経路と数値の組は、本発明の制御データ生成装置２０の出力である。

ー特に、上記経路は、本発明の制御データ生成装置２０における境界集合計算部２０８が出力した経路である。

ー特に、上記経路に対応付けられた数値は、当該経路に

おけるリボケーション関数値である。

【０１４２】次に利用デバイスについて説明する。

【０１４３】利用デバイスは、データを利用する為にマスター鍵を必要とする。それは、例えばデータがデータ・キーで暗号化されており、更にデータ・キーがマスター鍵で暗号化されているからである。あるいは、マスター鍵でデータが暗号化されている場合も考えられる。各利用デバイスは、デバイス情報を割り当てられている。デバイス情報は、本実施例のデバイス情報生成装置の出力である。即ち、デバイスIDと利用デバイスは制御データを読み込み、リボケーション関数値を復号して、マスター鍵を取得しようとする。

【０１４４】図２３に利用デバイスの構成を示す。

【０１４５】利用デバイスの動作を、図２４、図２５に示し以下に説明する。

【０１４６】まず、制御データ入力部に制御データ入力する（Ｓ１８０１）。ＣＰＵは制御データを制御データ入力部から読み取る（Ｓ１８０２）。ＣＰＵは制御データを制御データ格納部に格納する（Ｓ１８０３）。ＣＰＵは、デバイスIDの長さをデバイス情報格納部から読み、変数Nに設定する（Ｓ１８０４）。ここで、Ｊに１を初期設定する（Ｓ１８０５）。

【０１４７】この後、リボーク判定処理を行う。

【０１４８】まず、ＪはＮより大きいか否かを判別する（Ｓ１８０６）。大きければ、リボークであり、ＣＰＵは、メッセージ表示部に当該利用デバイスのリボークを表示する（Ｓ１８１１）。一方、現段階でＪはＮより大きくなければ、次の処理を行う。

・CPUが、デバイスIDの部分経路(P[１], , P[Ｊ])をデバイス情報格納部から読み出す（Ｓ１８０７）。

・CPUが、経路(P[１], , P[Ｊ])を制御情報格納部から検索する（Ｓ１８０８）。

・検索した結果、当該経路が存在しない場合には、Ｊを１だけ増分し、ステップＳ１８０６に戻る（Ｓ１８０９、Ｓ１８１０）。

【０１４９】ステップＳ１８０９にて当該経路が存在したと判断されると、ＣＰＵは、経路(P[１], , P[Ｊ])におけるリボケーション関数値RV[Ｊ]を読む（Ｓ１８１２）。次に、ＣＰＵは、J番目の経路関数値PF[Ｊ]をデバイス情報格納部から読む（Ｓ１８１３）。そして、ＣＰＵは、数の組(PF[Ｊ], RV[Ｊ])を復号部に送る（Ｓ１８１４）。

【０１５０】復号部では、PF[Ｊ]でRV[Ｊ]を復号し、得られた結果RをＣＰＵに返す（Ｓ１８１５）。ＣＰＵは、Rをデータ利用部に送る（Ｓ１８１６）。ＣＰＵは、データをデータ入力部から読む（Ｓ１８１７）。ＣＰＵは、データをデータ利用部に送る（Ｓ１８１８）。そして、データ利用部がデータを利用することとなる（Ｓ１８１９）。

【０１５１】以上の動作を解説する。

【０１５２】デバイス情報格納部３０３には、本実施例のデバイス情報生成装置１０が生成したデバイス情報の一組が格納されている。即ち、デバイスIDと、経路関数値の組である。

デバイスID：(P[１], P[２], P[３])

経路関数値：(PF1, PF2, PF3)

デバイス情報格納部３０３は、このデバイス情報に加えて、デバイスIDの長さを格納している。本実施例の場合、デバイスIDの長さは3である。

【０１５３】利用デバイス３０は、データ利用に先立って、制御データを読み込み、制御データ格納部３０２に格納する。ＣＰＵ３０７は、デバイスIDの部分経路を順に取り出し、制御データに含まれる経路の中で当該経路に一致するものがあるかを調べる。一致する経路が存在すれば、当該経路に対応するリボケーション関数値を制御データ格納部３０２から読み出す。

【０１５４】次いで、ＣＰＵ３０７は、デバイス情報格納部３０３から当該部分経路に対応する経路関数値を読み出し、経路関数値とリボケーション関数値の組を復号部３０６に送る。

【０１５５】復号部３０６は、経路関数値を鍵として、リボケーション関数値を復号し、その結果RをＣＰＵ３０７に返す。Rの値は、リボーク主体がマスター鍵として定めた値Kに等しい。ＣＰＵ３０７は、データ利用部３０４にRの値を送り、次いでデータ入力部３０５から読みこんだデータを、データ利用部３０４に送る。

【０１５６】利用デバイス３０によって読みこまれた制御データは、例えば、次の様に制御データ格納部に格納される。

((０), a(０)) ((２, １), a(２, １)) ((２, ０, ０), a(２, ０, ０))
((２, ２), a(２, ２)) ((２, ０, ２), a(２, ０, ２))
((１, ０), a(１, ０)) ((１, ２), a(１, ２)) ･･･式１０

制御データ生成装置２０の説明で示したように、この制御データは、デバイスID (２, ０, １)と経路(１, １)とをリボークするものである。

【０１５７】今、利用デバイス３０が、デバイスID (１, ０, ２)を有していたとする。利用デバイス３０のＣＰＵ３０７は、先ず部分経路(１)を、制御データ格納部３０２に格納されている、式１０の制御データから探す。この経路は式１０の制御データ中に存在しない。従って、利用デバイス３０のＣＰＵ３０７は、次の部分経路(１, ０)を式１０の制御データ中から探す。この経路は存在する。利用デバイス３０のＣＰＵ３０７は、対応するリボケーション関数値a(１, ０)を制御データ格納部３０２から読み出し、当該関数値を経路関数値PF2 = PF(１, ０)と共に復号部に送る。復号部３０６は、数値

PF2を鍵として数値a(1, 0)を復号し、その結果RをＣＰＵ３０７に返す。先述の様に、a(1, 0)はEnc(PF(1, 0), K)に等しい。従って、復号結果Rはマスター鍵Kに等しい。データ利用部３０４は、Kの値を得て、データ利用を行う事ができる。

【０１５８】一方、利用デバイス３０が、例えば(1, 1, 2)と言うデバイスIDを有しているとする。

【０１５９】ＣＰＵ３０７は、先ず部分経路(1)を、式１０の制御データから探す。この経路は式１０の制御データ中に存在しない。従って、CPUは次の部分経路(1, 1)を探す。この経路もまた式１０の制御データ中には存在しない。ＣＰＵ３０７は、更に部分経路(1, 1, 2)を、式１０の制御データ中から探す。この経路もまた存在しない。部分経路の長さがデバイス長に一致するまで検索を行ったので、ＣＰＵ３０７は当該利用デバイス３０がリボークされている事をメッセージ表示部３０８に表示し、処理を終了する。

【０１６０】本実施例の利用デバイスは、以下の点に特徴がある。

1. デバイス情報格納部の特徴：デバイス情報格納部には、デバイスIDの一部または全部の数字をとって作られる部分経路に対応する数値である経路関数値が少なくとも一つ予め格納されている。或いは、デバイス情報格納部は、本実施例のデバイス情報生成装置が生成したデバイス情報を格納している。

2. 入力データの特徴：入力データは制御データを含む。制御データは、経路と当該経路に対応するリボケーション関数値を、少なくとも一つ含む。

3. 動作の特徴：

(ア) 入力された制御データに含まれる経路の中から、デバイスIDの一部または全部の数字を並べて得られる部分経路に一致する経路を検索する。

(イ) 上記3.(ア)の検索の結果一致する経路が見つからなかった場合、当該利用デバイス３０がリボークされていると判断し、リボーク時の処理を行う。

(ウ) 上記3.(ア)の検索の結果一致する経路が見つかった場合、制御データから当該経路に対応するリボケーション関数値を読み出し、デバイス情報格納部３０３に当該経路に対応づけて格納されている経路関数値を鍵として復号する。上記特徴は、いずれもＭＫＢ技術における利用機器には無いものである。

【０１６１】（第２の実施の形態）次に、第２の実施の形態を説明する。

【０１６２】第１の実施の形態における制御データ生成装置２０は、随伴頂点集合計算部２１４を具備しており、制御データを作る際に、リボーク対象経路の随伴頂点集合を実際に生成した。

【０１６３】しかし、制御データ生成の為には、任意の経路がある経路の随伴頂点集合に属するか否かを判定する随伴頂点集合判定部があれば十分である事が、制御データ生成装置２０における制御データの生成手続きを検討する事により分かる。この事を以下に説明する。

【０１６４】第１の実施の形態の制御データ生成装置２０の手続きaでは、境界集合格納部が格納する経路集合U'と、経路pの随伴頂点集合V(p)の差集合が計算される。具体的には、Uに含まれる経路qを順に選び、qがV(p)に含まれていない場合に限り、qを作業メモリに記録して行けば良い。即ち、必ずしも随伴頂点集合V(p)を実際に生成する必要は無く、qがV(p)に含まれるか否かが分かれば良い。

【０１６５】第１の実施の形態の制御データ生成装置２０の手続きcでは、随伴頂点集合格納部２１２が格納する経路集合Uと、経路pの境界集合V'(p)の差集合V'(p) -Uを求める。この差集合を求める際は、V'(p)の各経路がUに含まれるか否かが分かれば良い。その為には、必ずしもUの要素を全て生成する必要は無い。

【０１６６】リボーク対象経路がデバイスIDの長さより相当に短い場合、当該経路の随伴頂点集合は、既に述べた様に、かなり大きな集合になる。この集合を生成する手間を省くことは、制御データ生成装置２０の動作を効率化する上で重要である。この点を改良した制御データ生成装置４０の構成を図２６に示す。

【０１６７】また、制御データ生成装置４０の初回の動作を図２７乃至図２９に、更新時の動作を図３０乃至図３４に示し説明する。

【０１６８】まず、初回の動作であるが、基本的には制御データ生成装置２０の初回の動作（図１１乃至図１３）とほぼ同様な動作を行う。

【０１６９】違いは、図１３におけるステップＳ１４２１～Ｓ１４２３の処理が無い点、及びステップＳ１４２４が、Ｖを随伴頂点集合格納部に格納するのに対し、ステップＳ１９２１では、ｐを随伴頂点集合格納部に格納する点である。このように制御データ生成装置４０の初回の動作は、制御データ生成装置４０の初回の動作に比較し、簡略化可能となった。

【０１７０】次に、更新時の動作について説明する。更新時においても制御データ生成装置４０の更新時の動作と、類似している。従って、違いについてのみ以下に書き出す。

【０１７１】図３０に記載される動作に関しては、ステップ２００６において、制御データ生成装置２０では、ｐを随伴頂点集合計算部に設定していたのを、制御データ生成装置４０では、ｐを随伴頂点集合判定部に設定することとなる。また、ステップＳ１５０７、Ｓ１５０８に相当する処理が不要になっている。

【０１７２】図３１に記載される動作に関しては、ステップＳ２０１１「ＣＰＵがＵを随伴頂点集合判定部に設定する」点が追加されたのみである。

【０１７３】図３２及び図３３に記載される動作に関しては、制御データ生成装置２０の図１６及び図１７に示

される動作と同様な動作で実現される。

【０１７４】図３４に示される動作が制御データ生成装置２０での動作と異なっている。

【０１７５】すなわち、ＣＰＵは、作業メモリから経路ｐを読み出し（Ｓ２０３５）、随伴頂点集合格納部にｐが存在するか否かを調べる（Ｓ２０３６）。ｐが存在すれば、そのまま終了し、ｐが存在しなければ、ｐを随伴頂点集合格納部に追加し（Ｓ２０３８）、動作を終了する。

【０１７６】随伴頂点集合判定部４１６は、図３０に示されている動作において、差集合U'－Vを求める際に用いられる。ここにVは、リボーク対象経路pの随伴頂点集合を表す。

【０１７７】また、随伴頂点集合判定部４１６は、図３１に示されている動作において、差集合V'－Uを求める際にも用いられる。

【０１７８】図３５は、差集合U'－Vを求める動作を説明したもので有り、図３６は差集合V'－Uを求める動作を説明したものである。これらの動作は、同様な動作を行うので、図３５に従って説明し、図３６の説明は、省略する。

【０１７９】まず、ＣＰＵがU'に属する経路の数を変数Ｎに設定する（Ｓ２１０１）。また、Ｊを1に設定する（Ｓ２１０２）。

【０１８０】その後、ＪがＮより大きくなるまでの次の処理を繰り返し行う。

・ＣＰＵは、U'のＪ番目の経路q[J]を随伴頂点集合判定部に送る（Ｓ２１０４）。

・随伴頂点集合判定部は、判定動作を行って、判定結果ＲをＣＰＵへ送る（Ｓ２１０５）。

・この判定結果Ｒが０でない場合のみ、ＣＰＵは、経路q[J]をU'－Ｖの経路として、作業メモリに格納する（Ｓ２１０７）。

・Ｊを１だけ増分する（Ｓ２１０８）。

【０１８１】次に、図３７及び図３８に随伴頂点集合判定部４１６の動作を示す。随伴頂点集合判定部４１６の動作は、経路集合設定動作と、判定動作に分けられる。

【０１８２】経路集合設定動作は、図３７の通り、経路集合Ｐを受け取り（Ｓ２３０１）、Ｐを経路集合格納部に格納する（Ｓ２３０２）。

【０１８３】判定動作は、図３８に示したように、まず、判定対象経路ｐを受け取る（Ｓ２４０１）。次に、経路集合格納部に格納されている経路集合Ｐの要素数を変数Ｎに設定する（Ｓ２４０２）。Ｊを１に設定する（Ｓ２４０３）。ＪがＮより大きくなるか（Ｓ２４０４）、ＩがＬより大きくなる（Ｓ２４０８）まで、以下の処理を行う。

・ＰのＪ番目の経路q[J]の長さと経路ｐの長さとのうち、大きくない方の長さを変数Ｌに設定する（Ｓ２４０６）。

・Ｉに1を設定する（Ｓ２４０７）。

・ＩはＬより大きいか否かを判定する（Ｓ２４０８）。

・経路q[J]とｐのＩ番目の数字を比較し（Ｓ２４０９）、一致するか否かを判定する（Ｓ２４１０）。

・一致した場合、Ｉを１増分し（Ｓ２４１１）、ステップＳ２４０８へ戻る。一致しなければ、Ｊを１増分し（Ｓ２４１２）、ステップＳ２４０４へ戻る。

【０１８４】ステップＳ２４０４において、ＪがＮより大きいと判断されると、０を出力し（Ｓ２４０５）、終了する。または、ステップＳ２４０８において、ＩがＬより大きいと判断されると、１を出力し（Ｓ２４１３）、終了する。

【０１８５】以上の判定動作を換言すると、判定動作は、判定対象経路を受け取り、当該経路が、経路集合格納部に格納されている経路の随伴頂点集合になっているかどうかを判定するものである。経路集合設定動作は、経路集合格納部の内容を設定する動作である。

【０１８６】二つの経路p，qを考える。経路qの長さが経路pの長さより長くない場合、qがpの随伴頂点集合に含まれる事と、qの数列全体がpの数列の一部に一致する事とは同値である。

【０１８７】経路qの長さが経路pの長さより長い場合は、pの数列全体がqの数列の一部に一致する事と、qがpの随伴頂点集合に含まれる事とは同値である。随伴頂点集合判定部４１６は、経路集合格納部４１７から経路qを順に選択し、qの数の並びと入力された経路の数の並びとを比較する事により、入力された経路がqの随伴頂点集合に含まれるか否かを判定している。

【０１８８】本実施例の制御データ生成装置４０は、随伴頂点集合判定部４１６を具備する点に特徴がある。

ー随伴頂点集合判定部４１６：数字の並びである経路を少なくとも一つ設定することができる。入力された経路が、設定された経路に属する数字の一部または全部を用いて作られる部分経路に一致するか否かを判定する。或いは、入力された経路が、設定された経路に少なくとも一つの数字を追加して作られる経路に一致するか否かを判定する。そして、判定結果を出力する。

【０１８９】

【発明の効果】本発明によれば、利用デバイスにおけるデータの利用を、制御データによって制御する技術において、重大なセキュリティ上の問題と、リボケーションに際して発生し、ユーザーの利便性を大きく損なう副作用の問題とを、共に回避する事が可能である。

【図面の簡単な説明】

【図１】リボケーション木構造の一

【図２】リボケーション木構造の二

【図３】リボケーション木構造の三

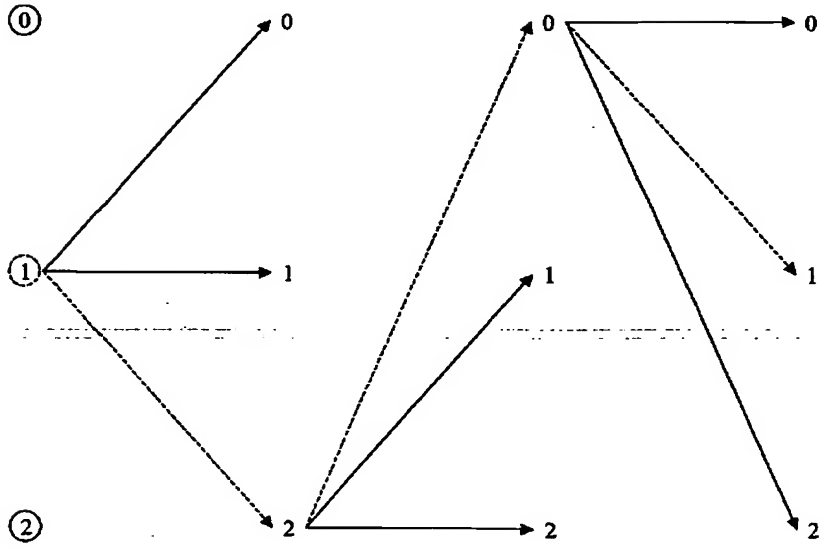
【図４】デバイス情報生成装置

【図５】経路関数計算部の動作

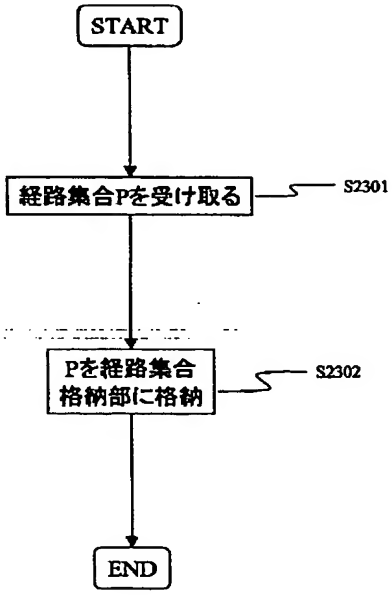
【図６】デバイス情報生成装置の動作の第一

【図７】デバイス情報生成装置の動作の第二
【図８】デバイス情報生成装置の動作の第三
【図９】制御データ生成装置の構成
【図１０】制御データ生成装置の動作（初期状態）
【図１１】制御データ生成装置の動作（初回）の第一
【図１２】制御データ生成装置の動作（初回）の第二
【図１３】制御データ生成装置の動作（初回）の第三
【図１４】制御データ生成装置の動作（更新時）の第一
【図１５】制御データ生成装置の動作（更新時）の第二
【図１６】制御データ生成装置の動作（更新時）の第三
【図１７】制御データ生成装置の動作（更新時）の第四
【図１８】制御データ生成装置の動作（更新時）の第五
【図１９】随伴頂点集合計算部の動作の第一
【図２０】随伴頂点集合計算部の動作の第二
【図２１】境界集合計算部の動作
【図２２】制御データのフォーマット
【図２３】利用デバイスの構成
【図２４】利用デバイスの動作の第一
【図２５】利用デバイスの動作の第二
【図２６】制御データ生成装置の構成
【図２７】制御データ生成装置の動作（初回）の第一
【図２８】制御データ生成装置の動作（初回）の第二
【図２９】制御データ生成装置の動作（初回）の第三
【図３０】制御データ生成装置の動作（更新時）の第一
【図３１】制御データ生成装置の動作（更新時）の第二
【図３２】制御データ生成装置の動作（更新時）の第三
【図３３】制御データ生成装置の動作（更新時）の第四
【図３４】制御データ生成装置の動作（更新時）の第五
【図３５】差集合U' － Vを求める動作
【図３６】差集合V' － Uを求める動作
【図３７】随伴頂点集合判定部の動作（設定時）
【図３８】随伴頂点集合判定部の動作（判定時）
【図３９】MKB処理
【図４０】デバイス情報生成装置
【図４１】デバイス情報生成装置の動作
【図４２】デバイス情報生成装置の動作
【図４３】MKB生成装置の構成
【図４４】メディア・キー・ブロック生成装置の動作（生成時）の第一
【図４５】メディア･キー･ブロック生成装置の動作（生成時）の第二
【図４６】メディア･キー･ブロック生成装置の動作（更新時）
【図４７】利用機器の構成
【図４８】利用機器の動作の第一
【図４９】利用機器の動作の第二
【符号の説明】
１０・・・デバイス情報生成装置、２０・・・制御データ生成装置
４０・・・制御データ生成装置

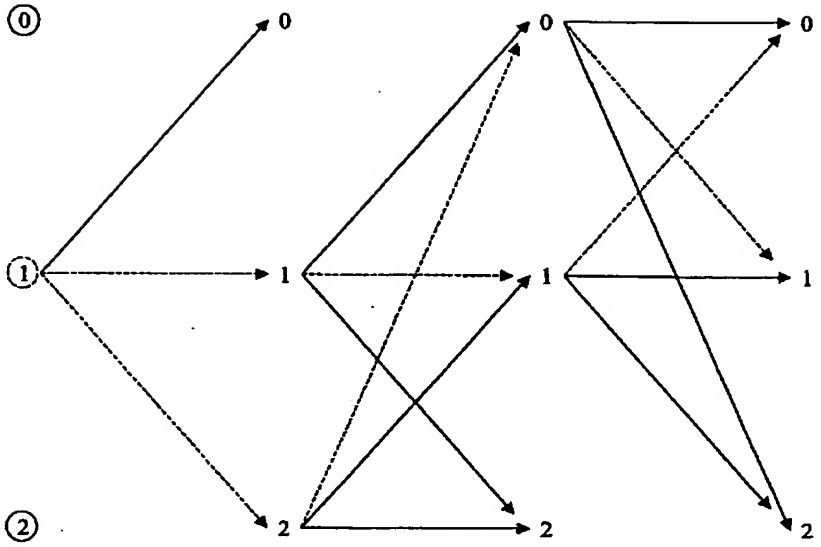
【図１】



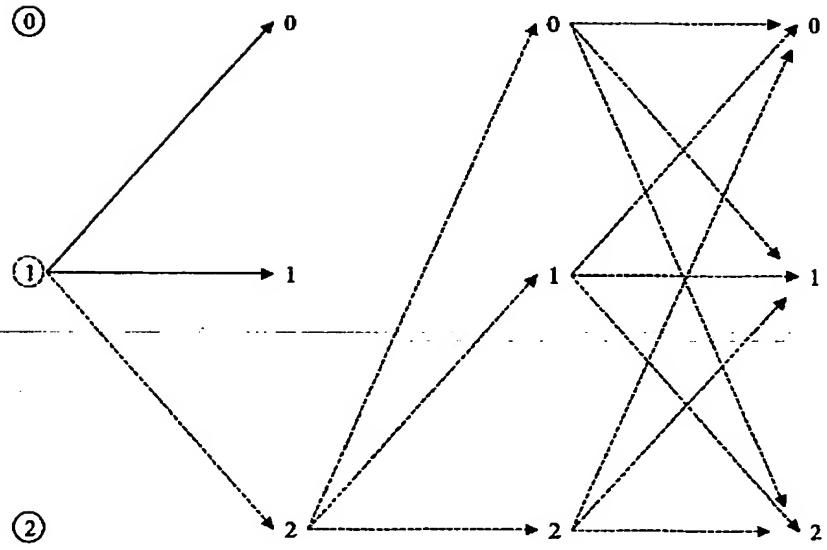
【図３７】



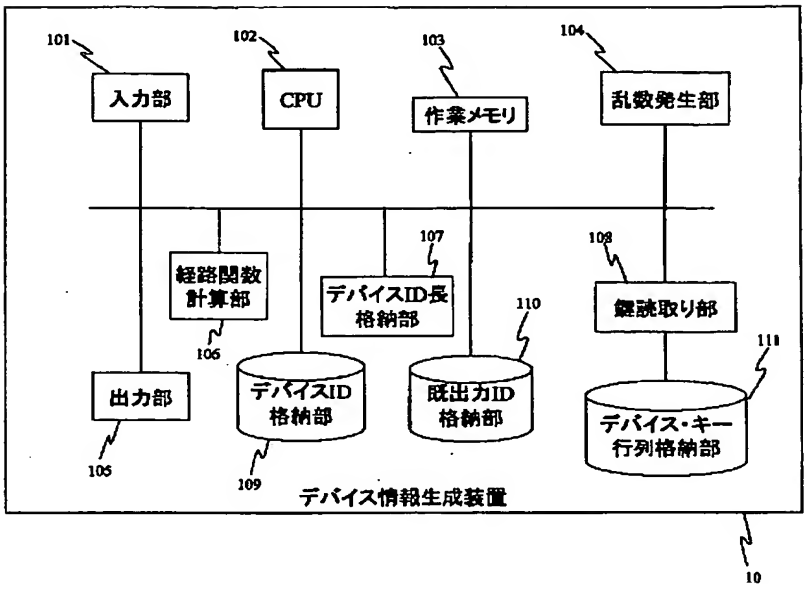
【図２】



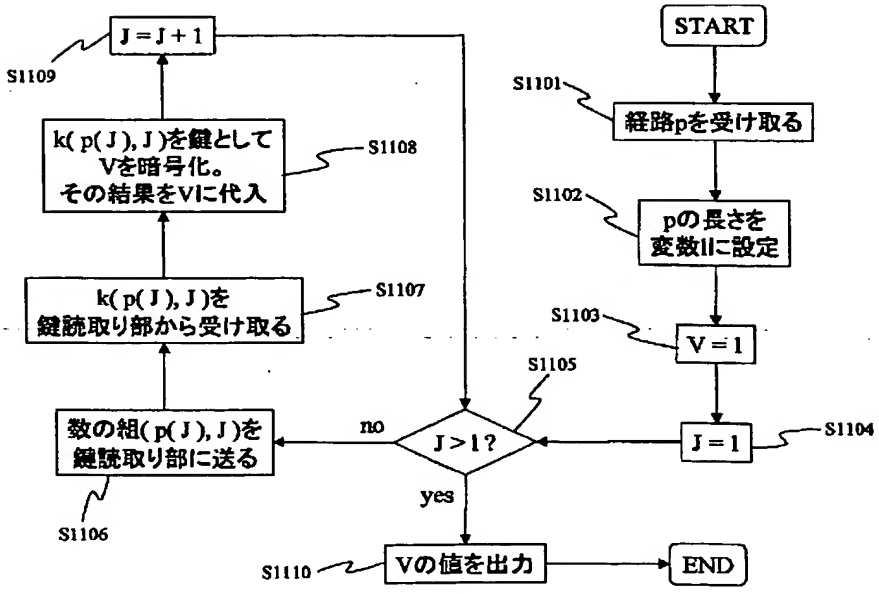
【図３】



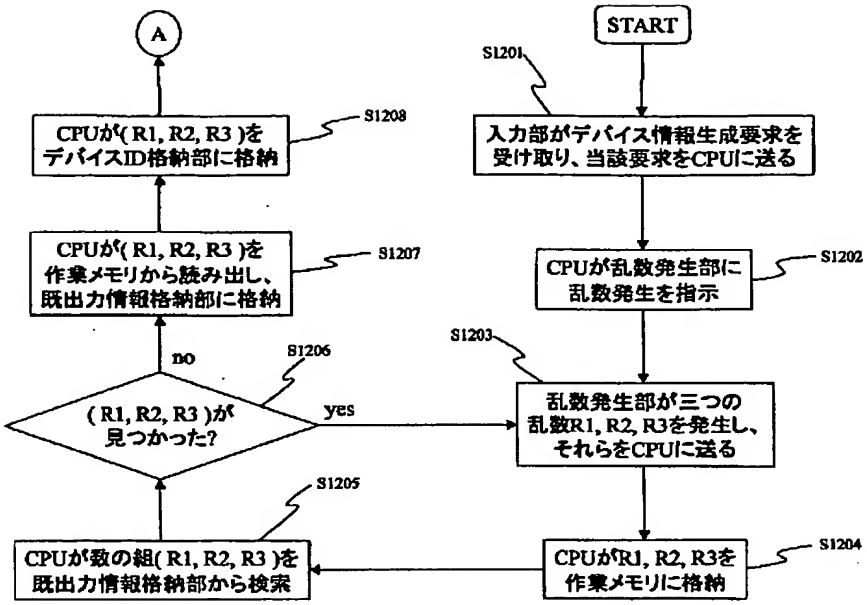
【図４】


101
入力部
102
CPU
103
作業メモリ
104
乱数発生部
経路関数計算部
106
107
デバイスID長格納部
108
鍵読取り部
110
111
出力部
105
デバイスID格納部
109
既出力ID格納部
デバイス・キー行列格納部
デバイス情報生成装置
10

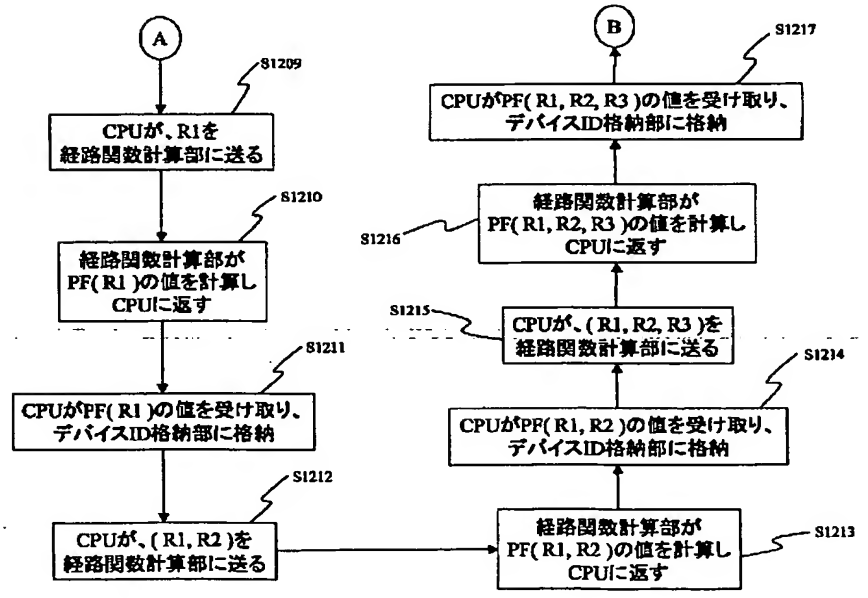
【図５】


START
S1101
経路pを受け取る
S1102
pの長さを変数lに設定
S1103
V＝1
S1104
J＝1
S1105
J＞l？
no
yes
S1106
数の組(p(J), J)を鍵読取り部に送る
S1107
k(p(J), J)を鍵読取り部から受け取る
S1108
k(p(J), J)を鍵としてVを暗号化。その結果をVに代入
S1109
J＝J＋1
S1110
Vの値を出力
END

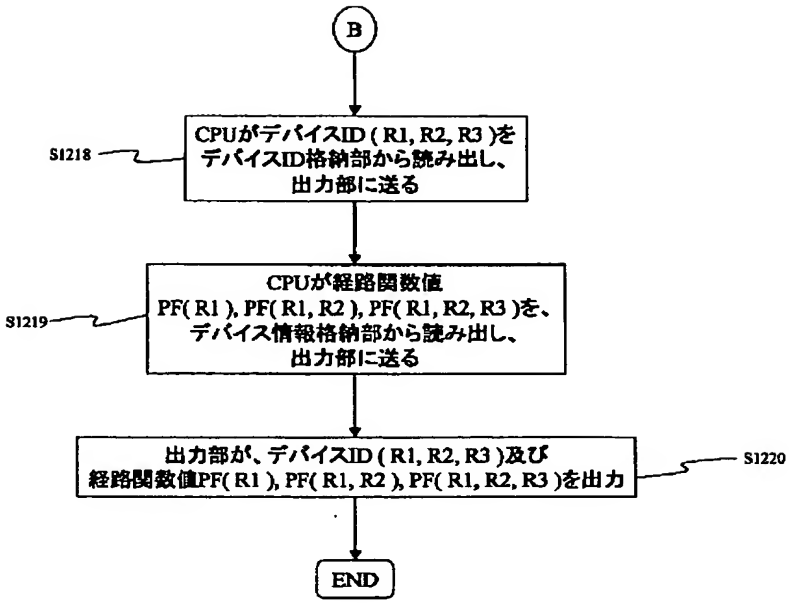
【図６】


START
S1201
入力部がデバイス情報生成要求を受け取り、当該要求をCPUに送る
S1202
CPUが乱数発生部に乱数発生を指示
S1203
乱数発生部が三つの乱数R1, R2, R3を発生し、それらをCPUに送る
S1204
CPUがR1, R2, R3を作業メモリに格納
S1205
CPUが数の組(R1, R2, R3)を既出力情報格納部から検索
S1206
(R1, R2, R3)が見つかった?
yes
no
S1207
CPUが(R1, R2, R3)を作業メモリから読み出し、既出力情報格納部に格納
S1208
CPUが(R1, R2, R3)をデバイスID格納部に格納
A

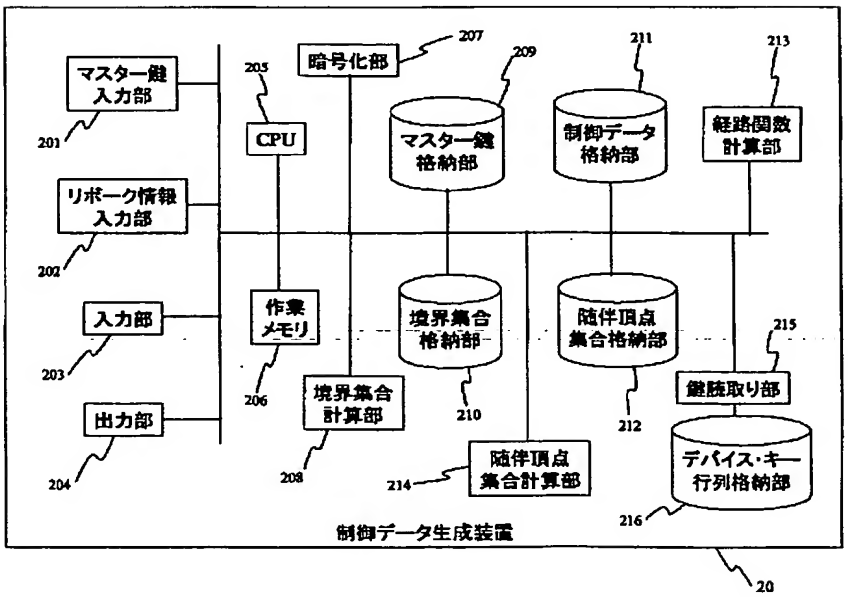
【図７】


A
S1209
CPUが、R1を経路関数計算部に送る
S1210
経路関数計算部がPF(R1)の値を計算しCPUに返す
S1211
CPUがPF(R1)の値を受け取り、デバイスID格納部に格納
S1212
CPUが、(R1, R2)を経路関数計算部に送る
S1213
経路関数計算部がPF(R1, R2)の値を計算しCPUに返す
S1214
CPUがPF(R1, R2)の値を受け取り、デバイスID格納部に格納
S1215
CPUが、(R1, R2, R3)を経路関数計算部に送る
S1216
経路関数計算部がPF(R1, R2, R3)の値を計算しCPUに返す
S1217
CPUがPF(R1, R2, R3)の値を受け取り、デバイスID格納部に格納
B

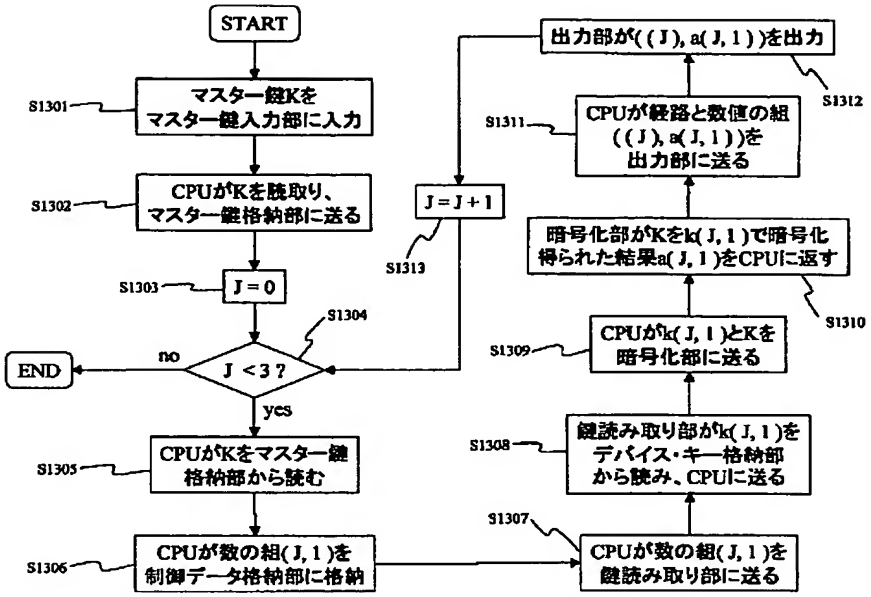
【図８】


B
S1218
CPUがデバイスID (R1, R2, R3)を
デバイスID格納部から読み出し、
出力部に送る
S1219
CPUが経路関数値
PF(R1), PF(R1, R2), PF(R1, R2, R3)を、
デバイス情報格納部から読み出し、
出力部に送る
出力部が、デバイスID (R1, R2, R3)及び
経路関数値PF(R1), PF(R1, R2), PF(R1, R2, R3)を出力
S1220
END

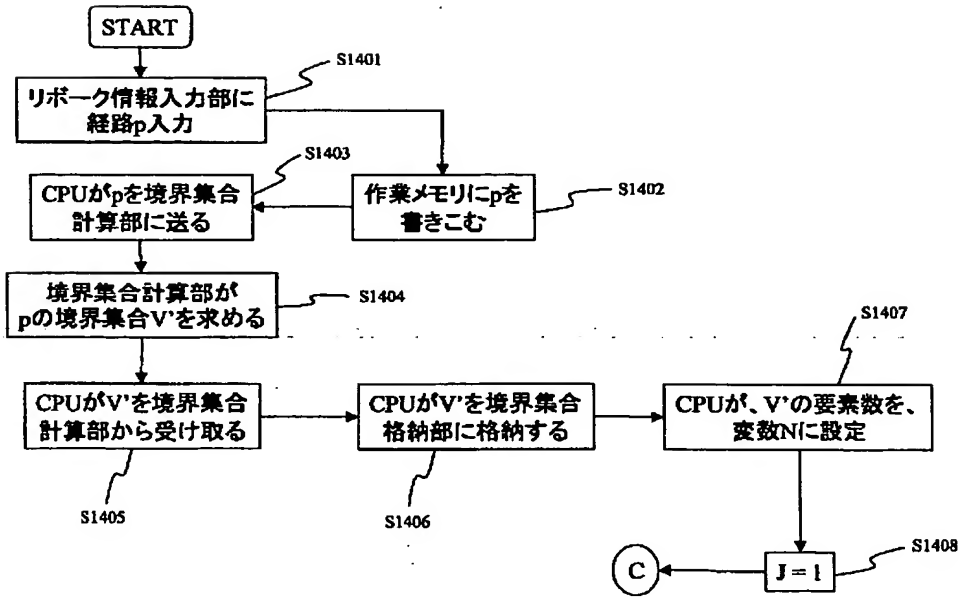
【図９】


マスター鍵
入力部
201
リポーク情報
入力部
202
入力部
203
出力部
204
CPU
205
作業
メモリ
206
暗号化部
207
境界集合
計算部
208
マスター鍵
格納部
209
境界集合
格納部
210
制御データ
格納部
211
随伴頂点
集合格納部
212
随伴頂点
集合計算部
214
経路関数
計算部
213
鍵読取り部
215
デバイス・キー
行列格納部
216
制御データ生成装置
20

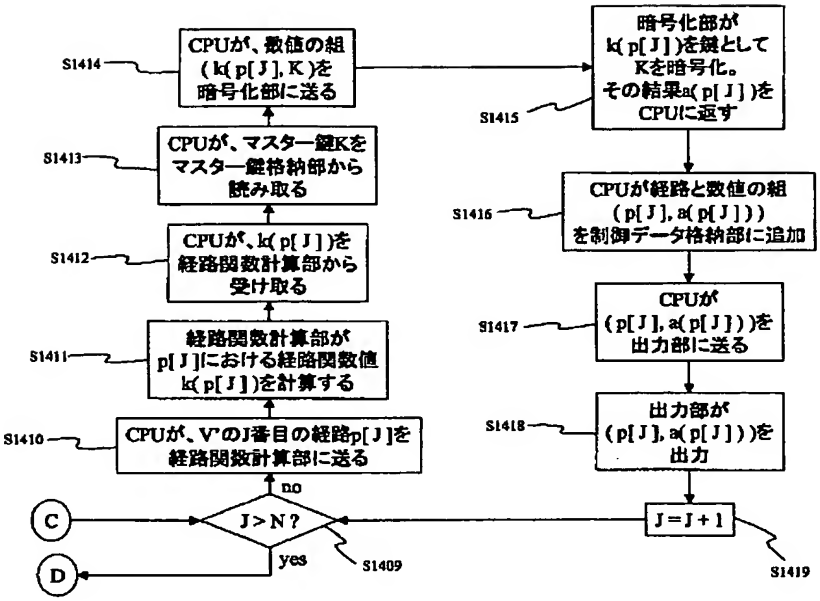
【図１０】


START
S1301
マスター鍵Kを
マスター鍵入力部に入力
S1302
CPUがKを読取り、
マスター鍵格納部に送る
S1303
J＝0
S1304
J＜3？
no
END
yes
S1305
CPUがKをマスター鍵
格納部から読む
S1306
CPUが数の組(J,1)を
制御データ格納部に格納
S1307
CPUが数の組(J,1)を
鍵読み取り部に送る
S1308
鍵読み取り部がk(J,1)を
デバイス・キー格納部
から読み、CPUに送る
S1309
CPUがk(J,1)とKを
暗号化部に送る
S1310
暗号化部がKをk(J,1)で暗号化
得られた結果a(J,1)をCPUに返す
S1311
CPUが経路と数値の組
((J),a(J,1))を
出力部に送る
S1312
出力部が((J),a(J,1))を出力
S1313
J＝J＋1

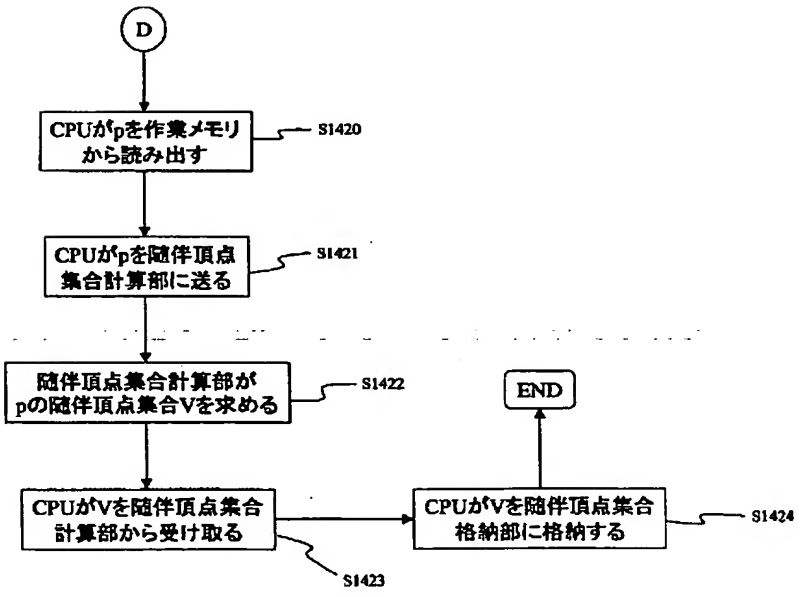
【図１１】


START
S1401
リボーク情報入力部に
経路p入力
S1402
作業メモリにpを
書きこむ
S1403
CPUがpを境界集合
計算部に送る
S1404
境界集合計算部が
pの境界集合V'を求める
S1405
CPUがV'を境界集合
計算部から受け取る
S1406
CPUがV'を境界集合
格納部に格納する
S1407
CPUが、V'の要素数を、
変数Nに設定
S1408
J＝1
C

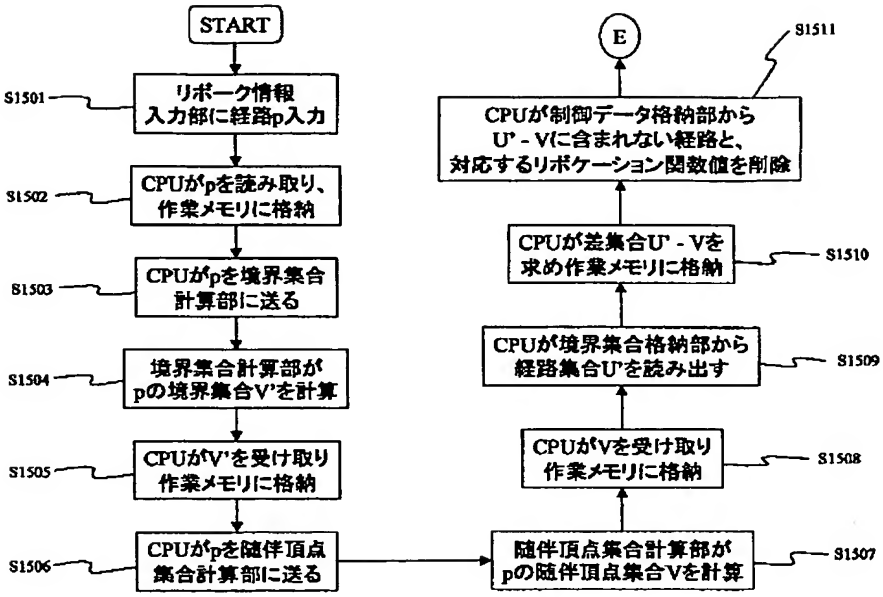
【図１２】


CPUが、最後の組(k(p[J],K)を暗号化部に送る
S1414
暗号化部がk(p[J])を鍵としてKを暗号化。その結果a(p[J])をCPUに返す
S1415
CPUが、マスター鍵Kをマスター鍵格納部から読み取る
S1413
CPUが経路と数値の組(p[J],a(p[J]))を制御データ格納部に追加
S1416
CPUが、k(p[J])を経路関数計算部から受け取る
S1412
CPUが(p[J],a(p[J]))を出力部に送る
S1417
経路関数計算部がp[J]における経路関数値k(p[J])を計算する
S1411
出力部が(p[J],a(p[J]))を出力
S1418
CPUが、V'のJ番目の経路p[J]を経路関数計算部に送る
S1410
no
C
J>N?
J=J+1
S1409
S1419
yes
D

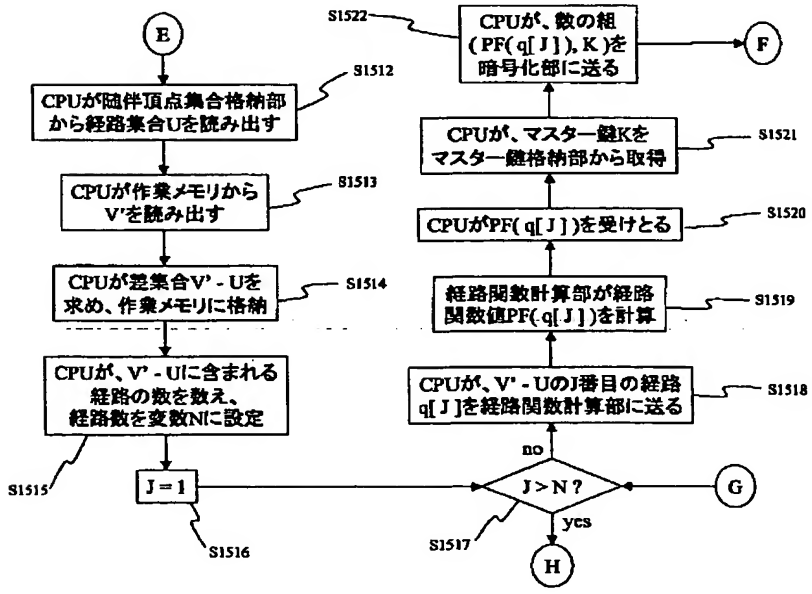
【図１３】


D
CPUがpを作業メモリから読み出す
S1420
CPUがpを随伴頂点集合計算部に送る
S1421
随伴頂点集合計算部がpの随伴頂点集合Vを求める
S1422
END
CPUがVを随伴頂点集合計算部から受け取る
S1423
CPUがVを随伴頂点集合格納部に格納する
S1424

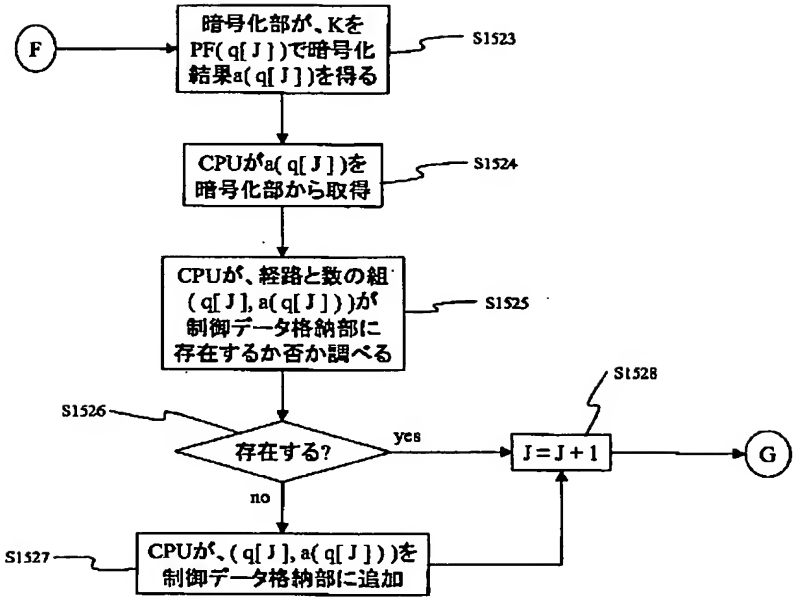
【図１４】


START
S1501
リポーク情報
入力部に経路p入力
S1502
CPUがpを読み取り、
作業メモリに格納
S1503
CPUがpを境界集合
計算部に送る
S1504
境界集合計算部が
pの境界集合V'を計算
S1505
CPUがV'を受け取り
作業メモリに格納
S1506
CPUがpを随伴頂点
集合計算部に送る
S1507
随伴頂点集合計算部が
pの随伴頂点集合Vを計算
S1508
CPUがVを受け取り
作業メモリに格納
S1509
CPUが境界集合格納部から
経路集合U'を読み出す
S1510
CPUが差集合U' - Vを
求め作業メモリに格納
S1511
CPUが制御データ格納部から
U' - Vに含まれない経路と、
対応するリポケーション関数値を削除
E

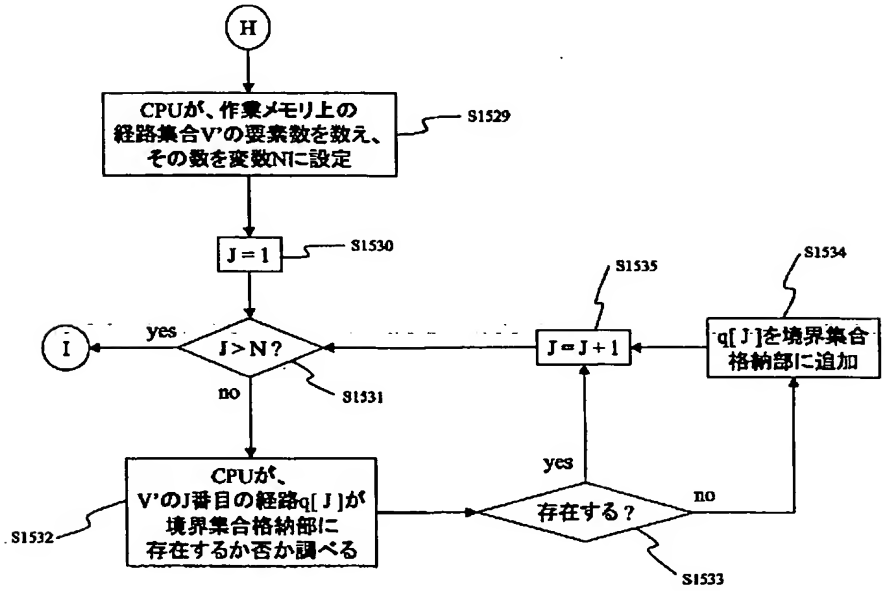
【図１５】


E
S1512
CPUが随伴頂点集合格納部
から経路集合Uを読み出す
S1513
CPUが作業メモリから
V'を読み出す
S1514
CPUが差集合V' - Uを
求め、作業メモリに格納
CPUが、V' - Uに含まれる
経路の数を数え、
経路数を変数Nに設定
S1515
J = 1
S1516
no
J > N ?
yes
S1517
G
H
S1518
CPUが、V' - UのJ番目の経路
q[J]を経路関数計算部に送る
S1519
経路関数計算部が経路
関数値PF(q[J])を計算
S1520
CPUがPF(q[J])を受けとる
S1521
CPUが、マスター鍵Kを
マスター鍵格納部から取得
S1522
CPUが、数の組
(PF(q[J]), K)を
暗号化部に送る
F

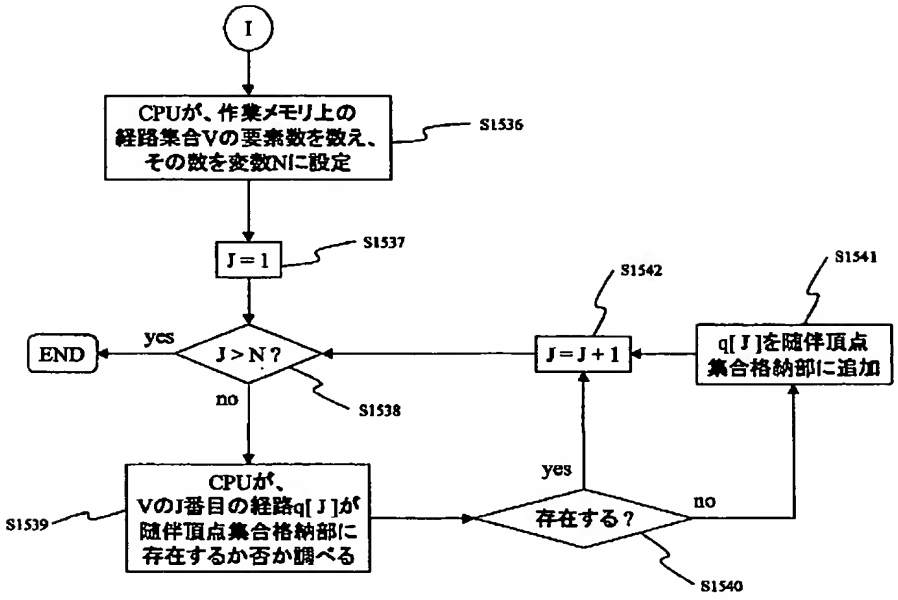
【図１６】


F
暗号化部が、Kを
PF(q[J])で暗号化
結果a(q[J])を得る
S1523
CPUがa(q[J])を
暗号化部から取得
S1524
CPUが、経路と数の組
(q[J], a(q[J]))が
制御データ格納部に
存在するか否か調べる
S1525
S1526
存在する?
yes
J=J+1
S1528
G
no
S1527
CPUが、(q[J], a(q[J]))を
制御データ格納部に追加

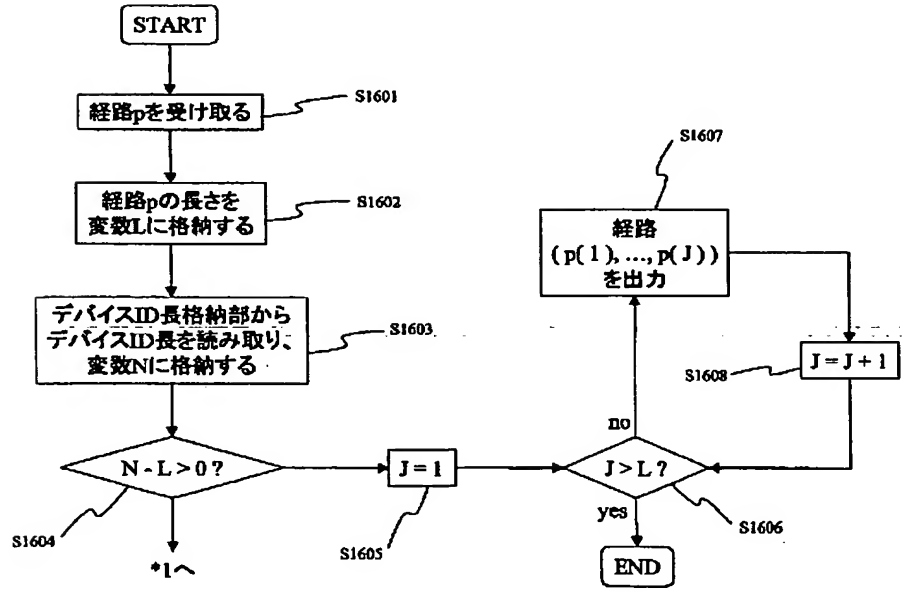
【図１７】


H
CPUが、作業メモリ上の
経路集合V'の要素数を数え、
その数を変数Nに設定
S1529
J=1
S1530
S1535
S1534
I
yes
J>N?
J=J+1
q[J]を境界集合
格納部に追加
S1531
no
yes
CPUが、
V'のJ番目の経路q[J]が
境界集合格納部に
存在するか否か調べる
存在する?
no
S1532
S1533

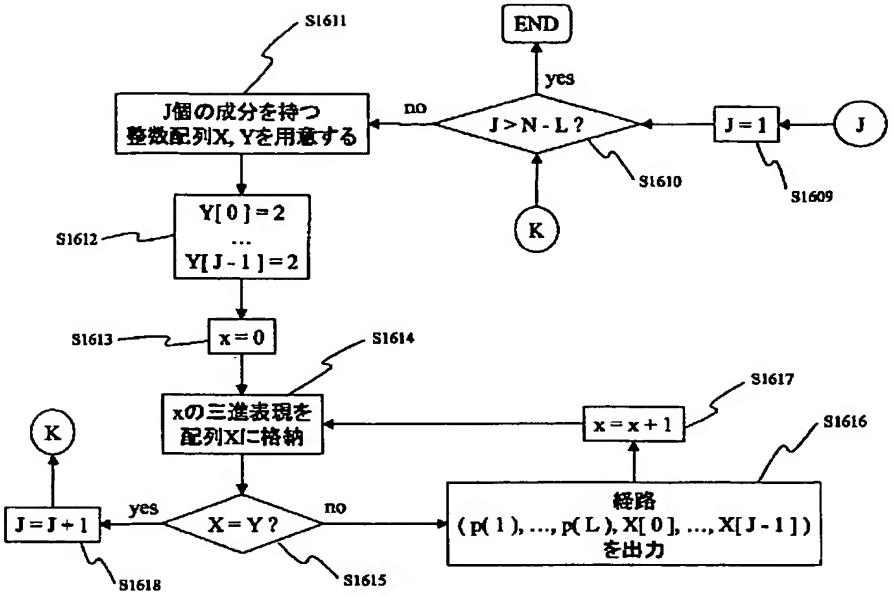
【図１８】


I
CPUが、作業メモリ上の経路集合Vの要素数を数え、その数を変数Nに設定
S1536
J＝1
S1537
J＞N？
S1538
yes
END
no
J＝J＋1
S1542
q[J]を随伴頂点集合格納部に追加
S1541
CPUが、VのJ番目の経路q[J]が随伴頂点集合格納部に存在するか否か調べる
S1539
存在する？
S1540
yes
no

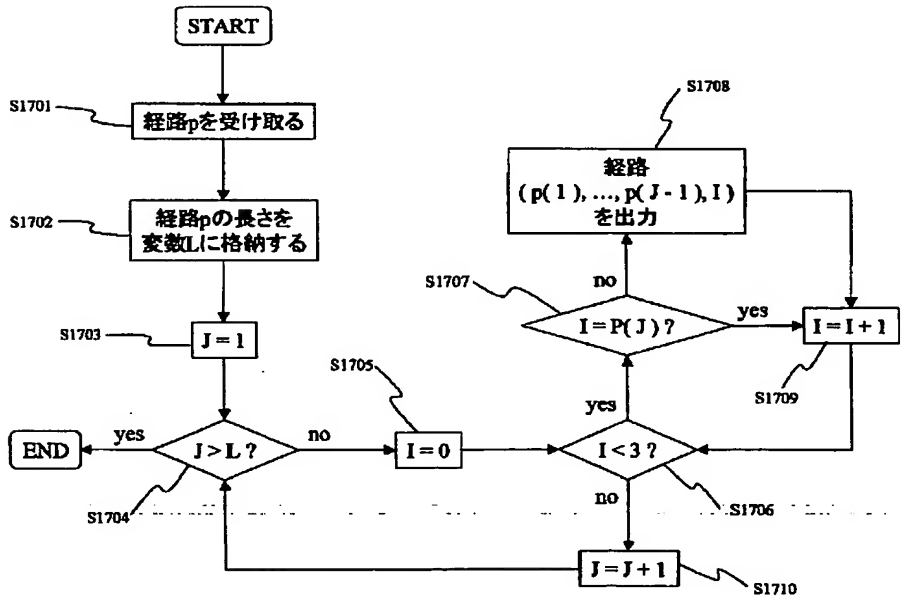
【図１９】


START
経路pを受け取る
S1601
経路pの長さを変数Lに格納する
S1602
デバイスID長格納部からデバイスID長を読み取り、変数Nに格納する
S1603
N－L＞0？
S1604
*1へ
J＝1
S1605
J＞L？
S1606
no
yes
END
経路(p(1),…,p(J))を出力
S1607
J＝J＋1
S1608

【図２０】



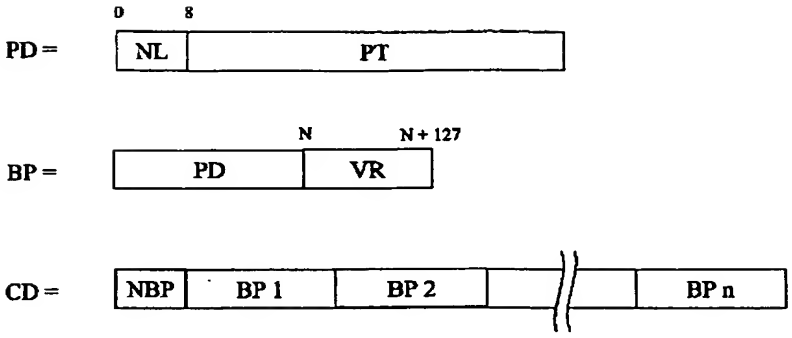
【図２１】



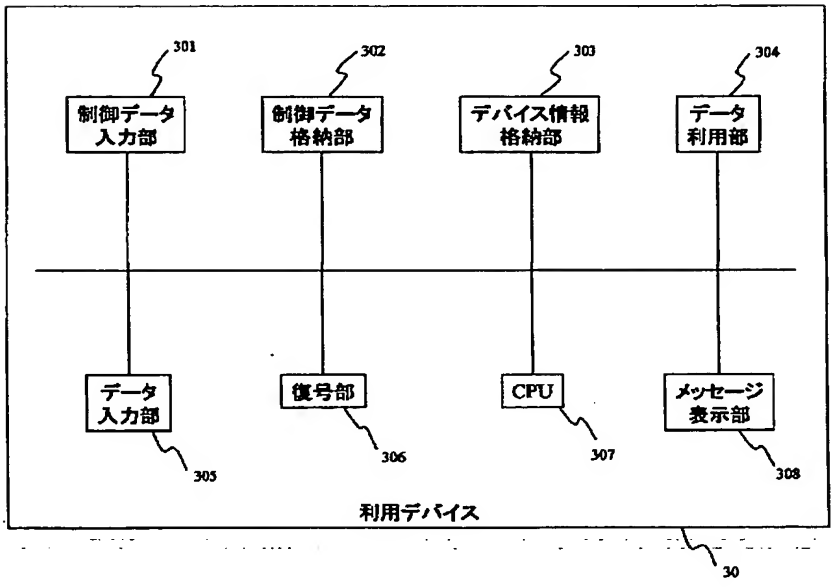
【図３９】

```
int         P[n];
NNum        KDP[n];

int         j;
NNum        Result;

for (j=0;j<n;j++)
{
            Result = Dec( KD[p[j],j], M[p[j],j] );
            if ( Result != null )
                        break;
}
```

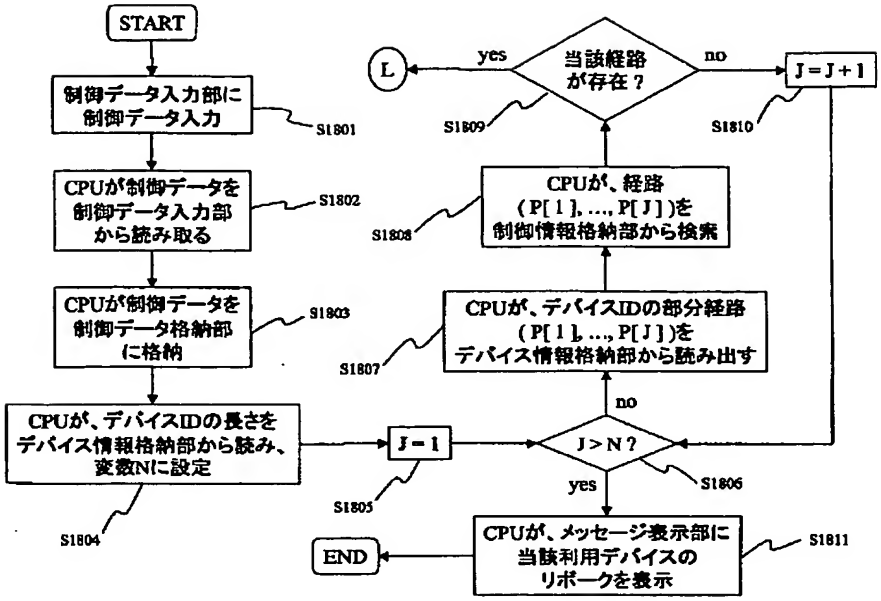
【図２２】


0
8
PD＝
NL
PT
N
N＋127
BP＝
PD
VR
CD＝
NBP
BP 1
BP 2
BP n

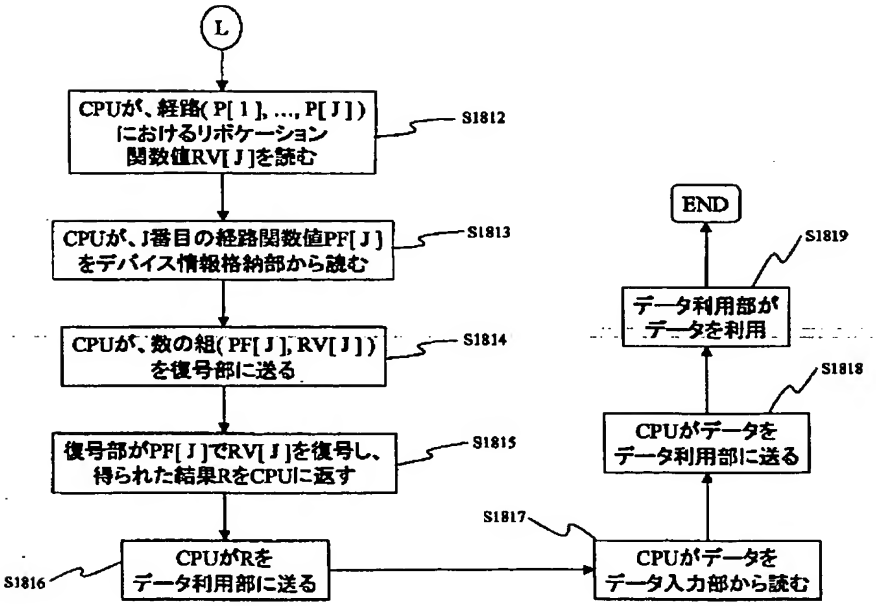
【図２３】


301
制御データ入力部
302
制御データ格納部
303
デバイス情報格納部
304
データ利用部
データ入力部
305
復号部
306
CPU
307
メッセージ表示部
308
利用デバイス
30

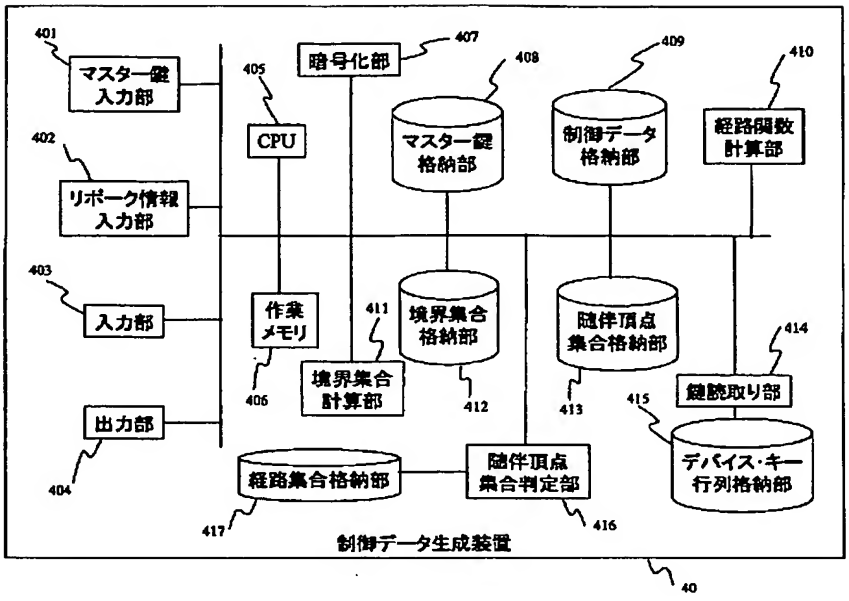
【図２４】


START
制御データ入力部に制御データ入力
S1801
CPUが制御データを制御データ入力部から読み取る
S1802
CPUが制御データを制御データ格納部に格納
S1803
CPUが、デバイスIDの長さをデバイス情報格納部から読み、変数Nに設定
S1804
J=1
S1805
J>N?
S1806
yes
no
CPUが、デバイスIDの部分経路(P[1],...,P[J])をデバイス情報格納部から読み出す
S1807
CPUが、経路(P[1],...,P[J])を制御情報格納部から検索
S1808
当該経路が存在?
S1809
yes
no
L
J=J+1
S1810
CPUが、メッセージ表示部に当該利用デバイスのリポークを表示
S1811
END

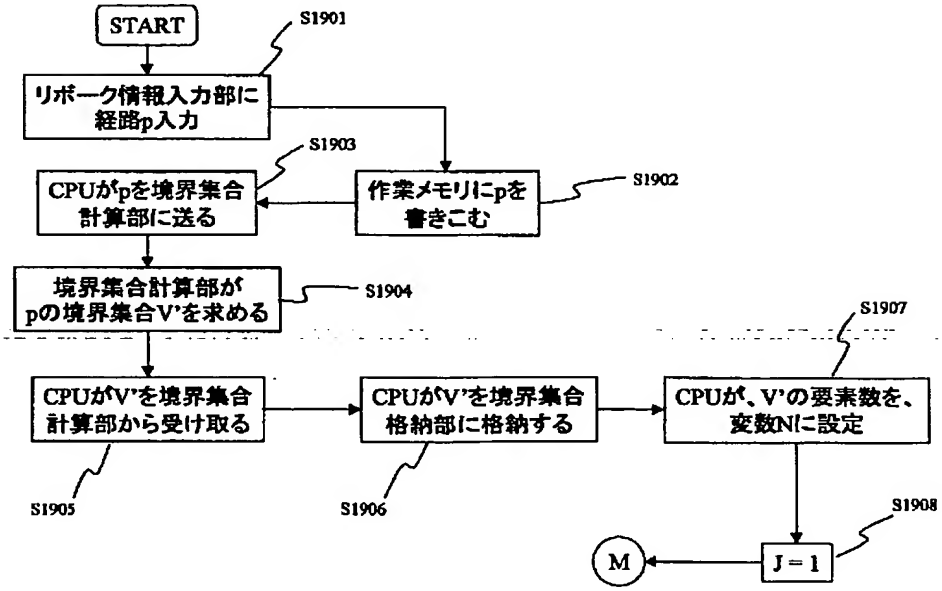
【図２５】


L
CPUが、経路(P[1],...,P[J])におけるリポケーション関数値RV[J]を読む
S1812
CPUが、J番目の経路関数値PF[J]をデバイス情報格納部から読む
S1813
CPUが、数の組(PF[J],RV[J])を復号部に送る
S1814
復号部がPF[J]でRV[J]を復号し、得られた結果RをCPUに返す
S1815
CPUがRをデータ利用部に送る
S1816
CPUがデータをデータ入力部から読む
S1817
CPUがデータをデータ利用部に送る
S1818
データ利用部がデータを利用
S1819
END

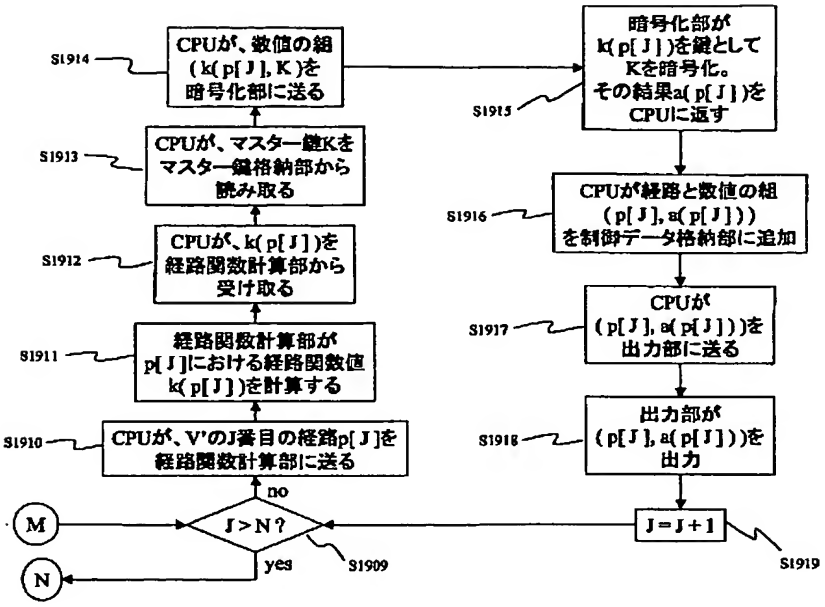
【図２６】


401
マスター鍵入力部
402
リポーク情報入力部
403
入力部
出力部
404
405
CPU
作業メモリ
406
暗号化部
407
境界集合計算部
408
マスター鍵格納部
411
境界集合格納部
412
409
制御データ格納部
410
経路個数計算部
随伴頂点集合格納部
413
414
鍵読取り部
415
デバイス・キー行列格納部
経路集合格納部
417
随伴頂点集合判定部
416
制御データ生成装置
40

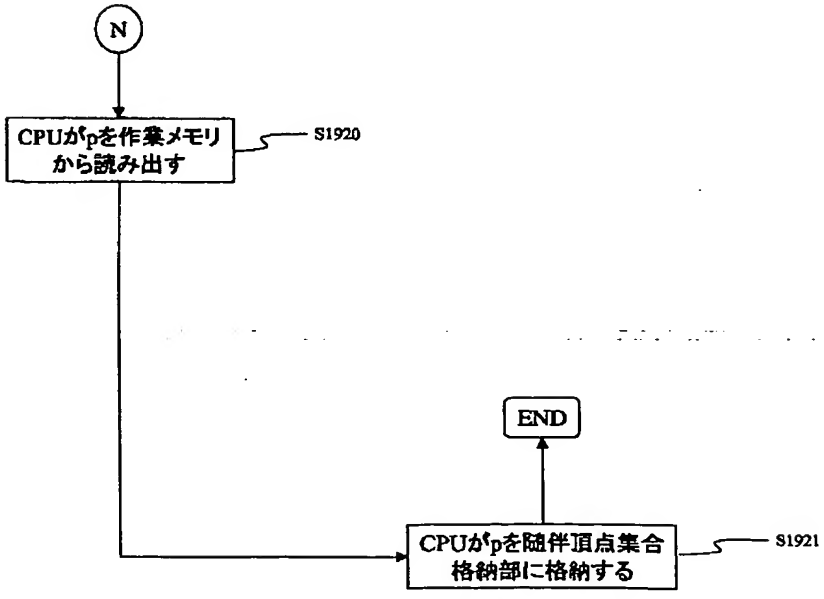
【図２７】


START
S1901
リポーク情報入力部に経路p入力
S1903
CPUがpを境界集合計算部に送る
作業メモリにpを書きこむ
S1902
境界集合計算部がpの境界集合V'を求める
S1904
S1907
CPUがV'を境界集合計算部から受け取る
CPUがV'を境界集合格納部に格納する
CPUが、V'の要素数を、変数Nに設定
S1905
S1906
S1908
M
J＝1

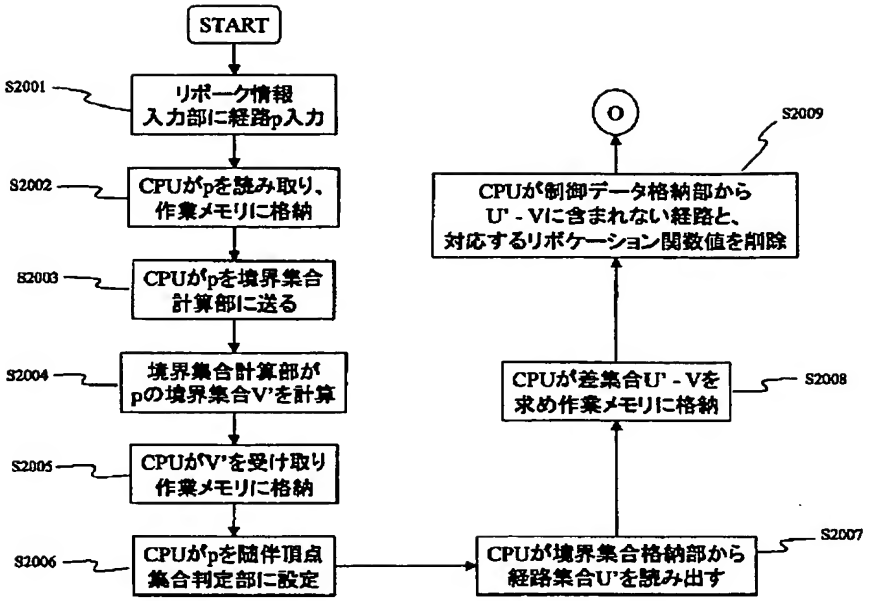
【図２８】


S1914
CPUが、数値の組
(k(p[J],K)を
暗号化部に送る
暗号化部が
k(p[J])を鍵として
Kを暗号化。
その結果a(p[J])を
CPUに返す
S1915
S1913
CPUが、マスター鍵Kを
マスター鍵格納部から
読み取る
CPUが経路と数値の組
(p[J],a(p[J]))
を制御データ格納部に追加
S1916
S1912
CPUが、k(p[J])を
経路関数計算部から
受け取る
CPUが
(p[J],a(p[J]))を
出力部に送る
S1917
S1911
経路関数計算部が
p[J]における経路関数値
k(p[J])を計算する
出力部が
(p[J],a(p[J]))を
出力
S1918
S1910
CPUが、V'のJ番目の経路p[J]を
経路関数計算部に送る
no
M
J>N?
J=J+1
S1909
S1919
yes
N

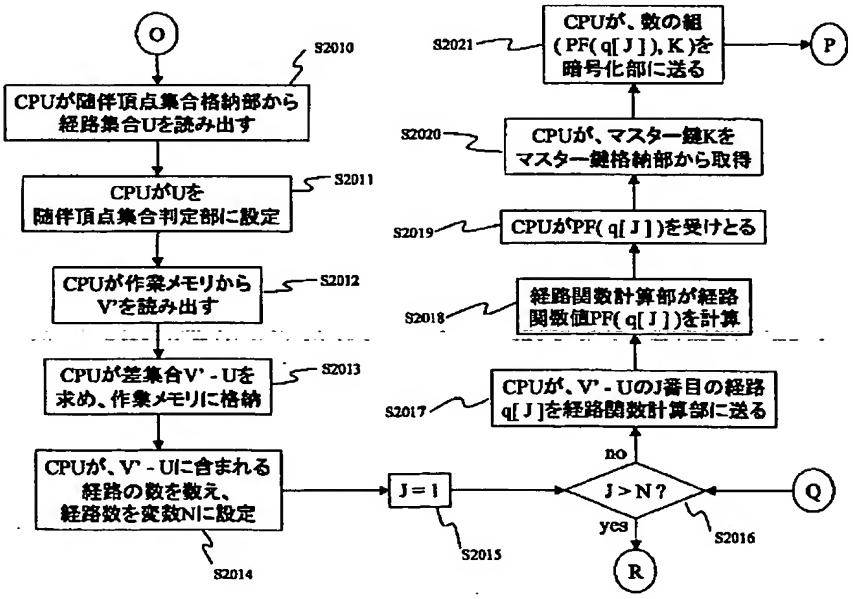
【図２９】


N
CPUがpを作業メモリ
から読み出す
S1920
END
CPUがpを随伴頂点集合
格納部に格納する
S1921

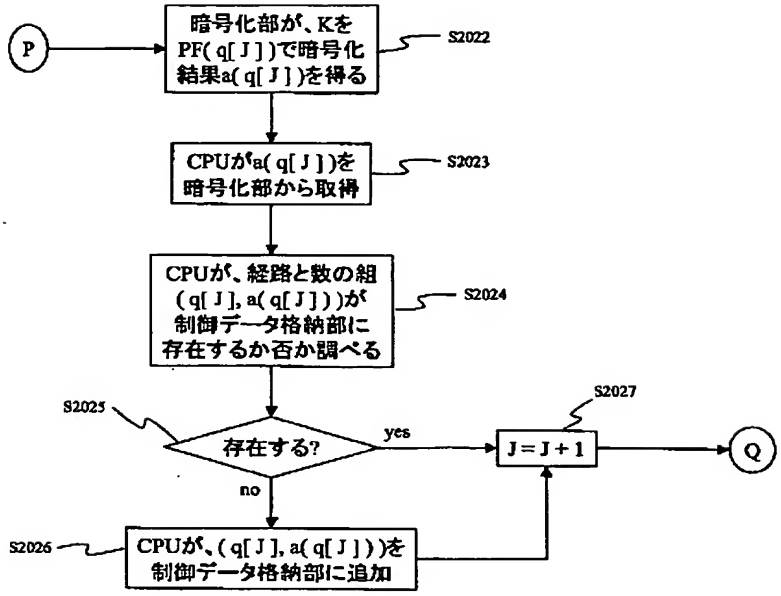
【図３０】


START
S2001
リポーク情報
入力部に経路p入力
S2002
CPUがpを読み取り、
作業メモリに格納
S2003
CPUがpを境界集合
計算部に送る
S2004
境界集合計算部が
pの境界集合V'を計算
S2005
CPUがV'を受け取り
作業メモリに格納
S2006
CPUがpを随伴頂点
集合判定部に設定
S2007
CPUが境界集合格納部から
経路集合U'を読み出す
S2008
CPUが差集合U' - Vを
求め作業メモリに格納
S2009
CPUが制御データ格納部から
U' - Vに含まれない経路と、
対応するリポケーション関数値を削除
O

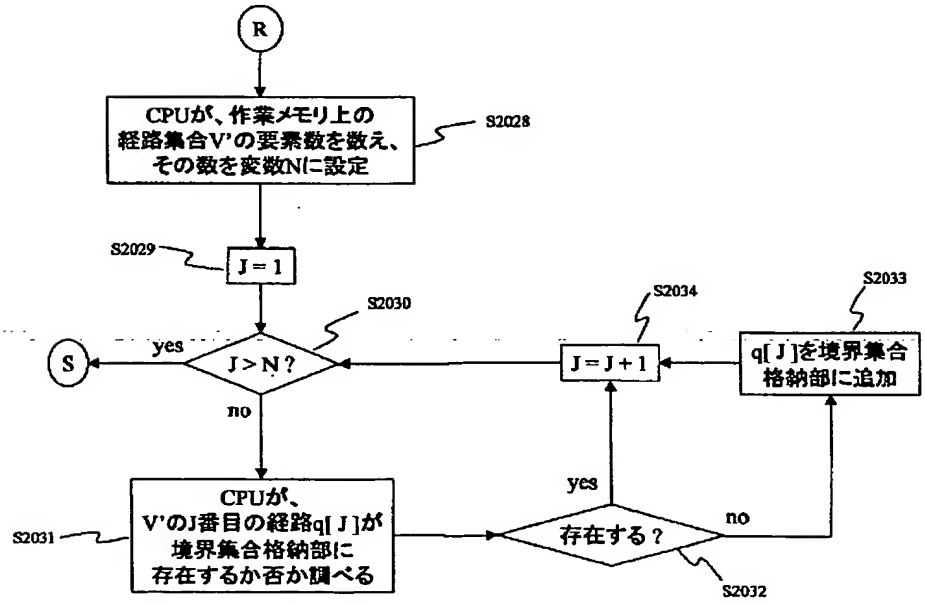
【図３１】


O
S2010
CPUが随伴頂点集合格納部から
経路集合Uを読み出す
S2011
CPUがUを
随伴頂点集合判定部に設定
S2012
CPUが作業メモリから
V'を読み出す
S2013
CPUが差集合V' - Uを
求め、作業メモリに格納
S2014
CPUが、V' - Uに含まれる
経路の数を数え、
経路数を変数Nに設定
J = 1
S2015
J > N ?
no
yes
S2016
Q
R
S2017
CPUが、V' - UのJ番目の経路
q[J]を経路関数計算部に送る
S2018
経路関数計算部が経路
関数値PF(q[J])を計算
S2019
CPUがPF(q[J])を受けとる
S2020
CPUが、マスター鍵Kを
マスター鍵格納部から取得
S2021
CPUが、数の組
(PF(q[J]), K)を
暗号化部に送る
P

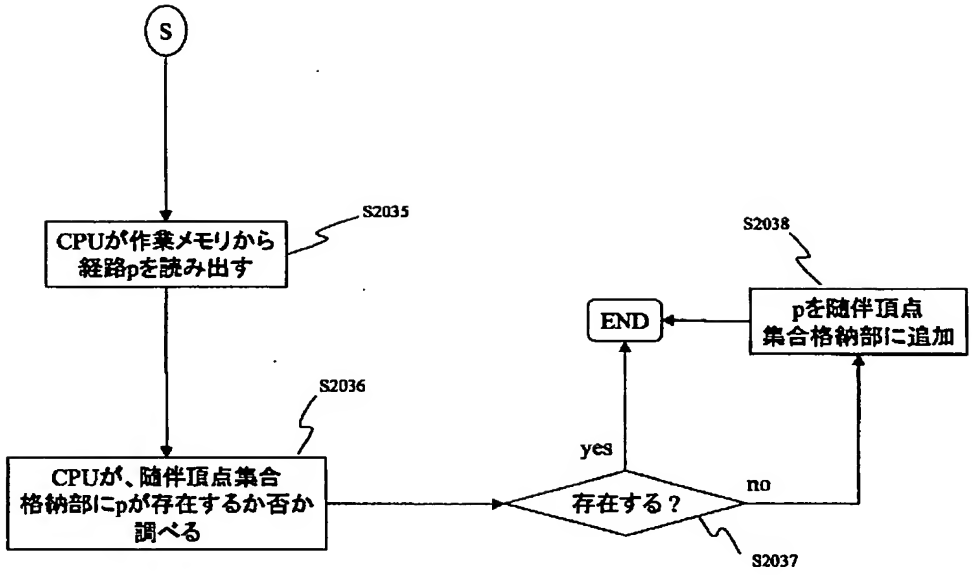
【図３２】


P
暗号化部が、Kを
PF(q[J])で暗号化
結果a(q[J])を得る
S2022
CPUがa(q[J])を
暗号化部から取得
S2023
CPUが、経路と数の組
(q[J], a(q[J]))が
制御データ格納部に
存在するか否か調べる
S2024
S2025
存在する?
yes
S2027
J = J + 1
Q
no
S2026
CPUが、(q[J], a(q[J]))を
制御データ格納部に追加

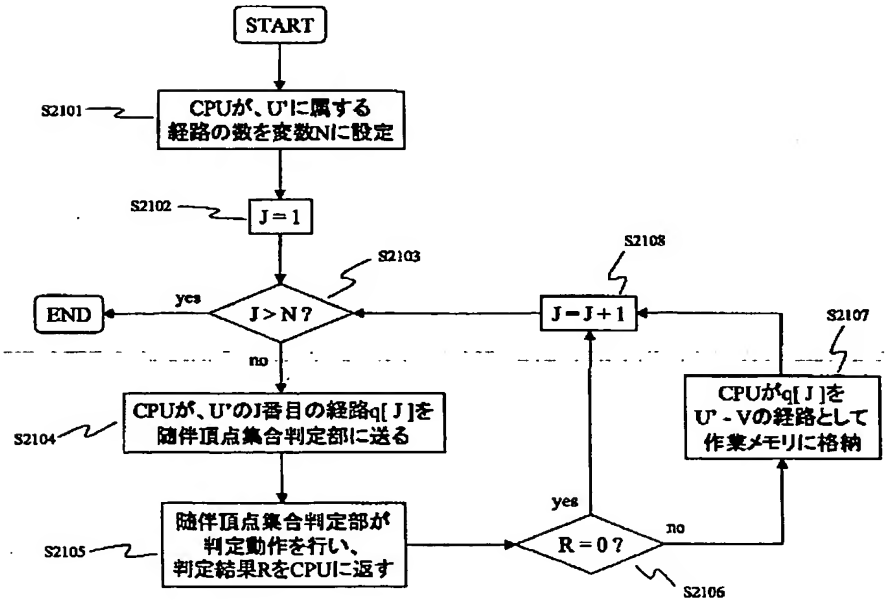
【図３３】


R
CPUが、作業メモリ上の
経路集合V'の要素数を数え、
その数を変数Nに設定
S2028
S2029
J = 1
S2030
S
yes
J > N ?
S2034
J = J + 1
S2033
q[J]を境界集合
格納部に追加
no
yes
S2031
CPUが、
V'のJ番目の経路q[J]が
境界集合格納部に
存在するか否か調べる
存在する ?
no
S2032

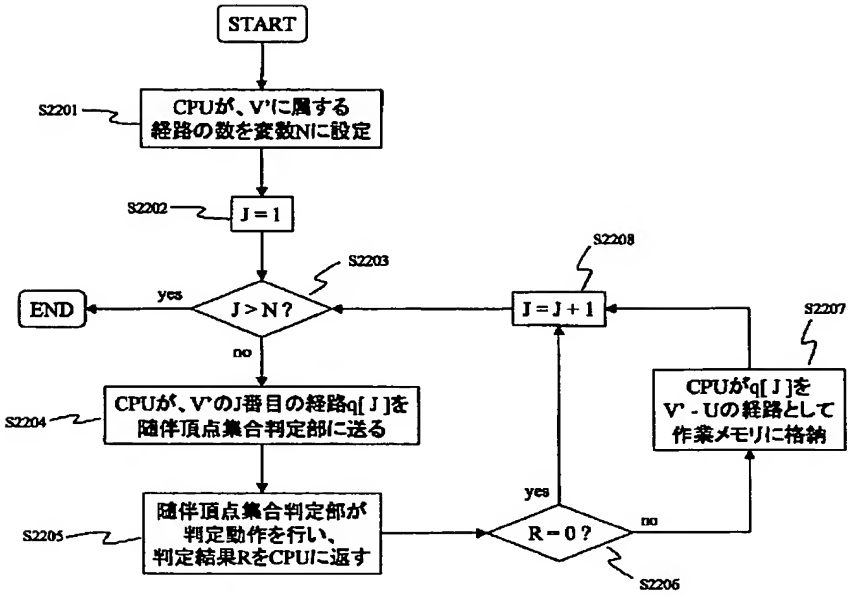
【図３４】


S
CPUが作業メモリから経路pを読み出す
S2035
CPUが、随伴頂点集合格納部にpが存在するか否か調べる
S2036
存在する？
yes
no
S2037
END
pを随伴頂点集合格納部に追加
S2038

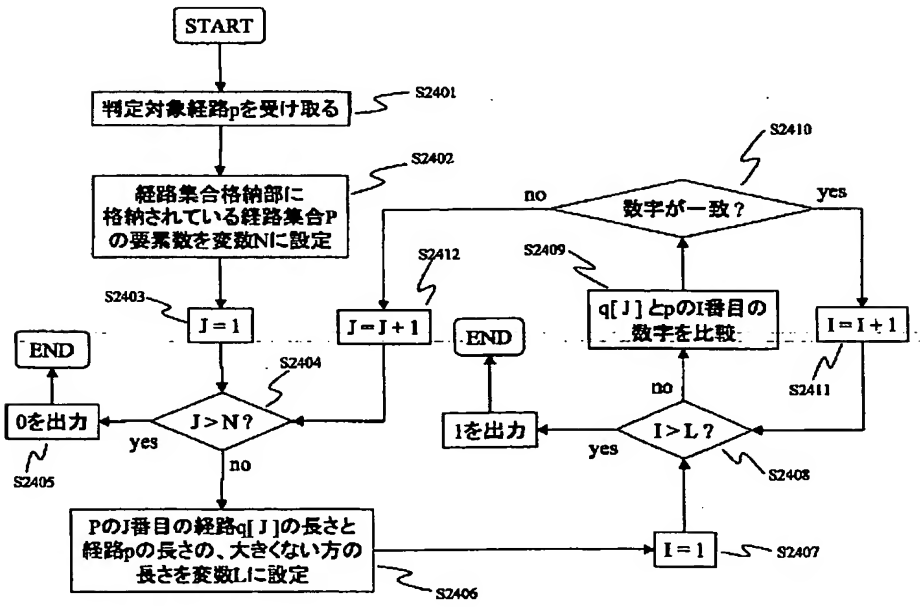
【図３５】


START
CPUが、U'に属する経路の数を変数Nに設定
S2101
J＝1
S2102
J＞N？
S2103
yes
no
END
CPUが、U'のJ番目の経路q[J]を随伴頂点集合判定部に送る
S2104
随伴頂点集合判定部が判定動作を行い、判定結果RをCPUに返す
S2105
R＝0？
yes
no
S2106
J＝J＋1
S2108
CPUがq[J]をU' - Vの経路として作業メモリに格納
S2107

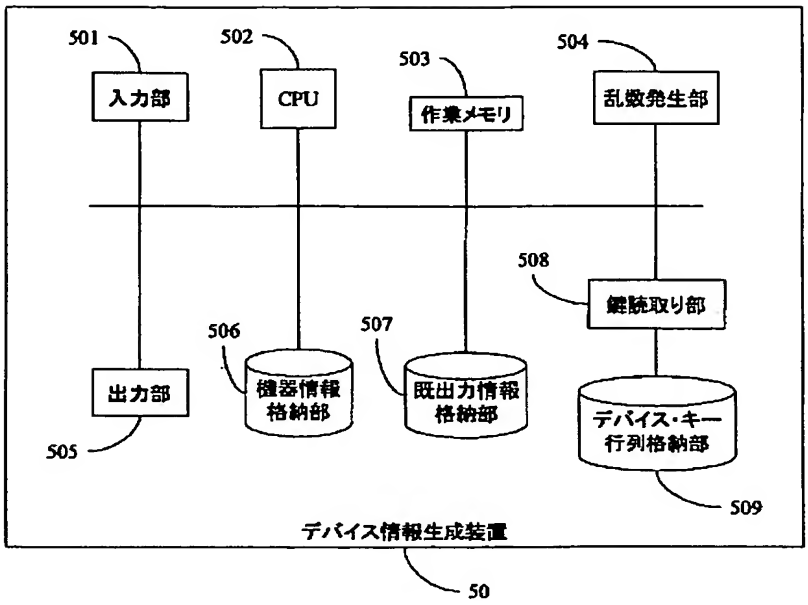
【図３６】


START
S2201
CPUが、V'に属する経路の数を変数Nに設定
S2202
J＝1
S2203
J＞N？
yes
END
no
S2208
J＝J＋1
S2207
CPUがq[J]をV'－Uの経路として作業メモリに格納
S2204
CPUが、V'のJ番目の経路q[J]を随伴頂点集合判定部に送る
S2205
随伴頂点集合判定部が判定動作を行い、判定結果RをCPUに返す
yes
R＝0？
no
S2206

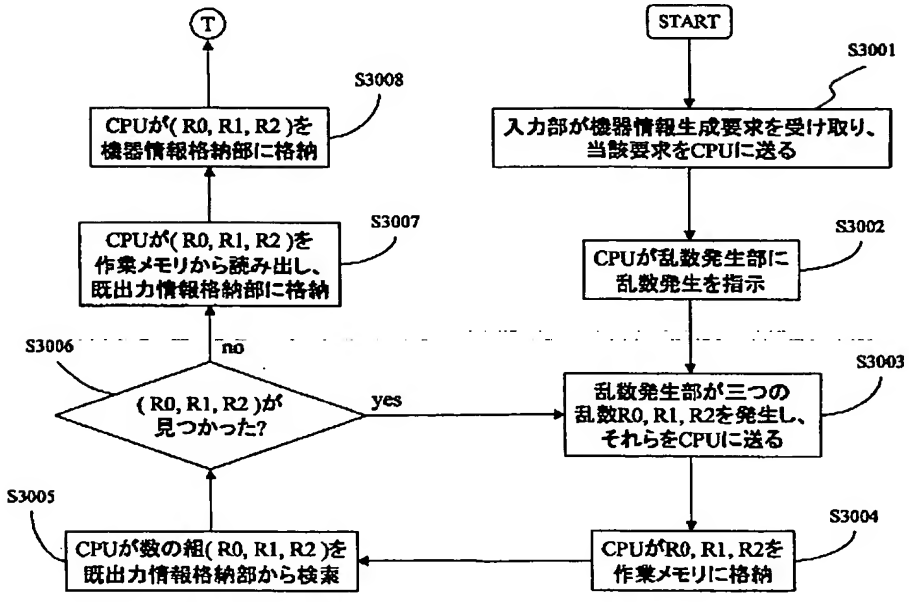
【図３８】


START
判定対象経路pを受け取る
S2401
経路集合格納部に格納されている経路集合Pの要素数を変数Nに設定
S2402
S2403
J＝1
END
0を出力
S2405
S2404
J＞N？
yes
no
PのJ番目の経路q[J]の長さと経路pの長さの、大きくない方の長さを変数Lに設定
S2406
I＝1
S2407
I＞L？
yes
no
S2408
1を出力
END
S2412
J＝J＋1
S2410
数字が一致？
no
yes
S2409
q[J]とpのI番目の数字を比較
I＝I＋1
S2411

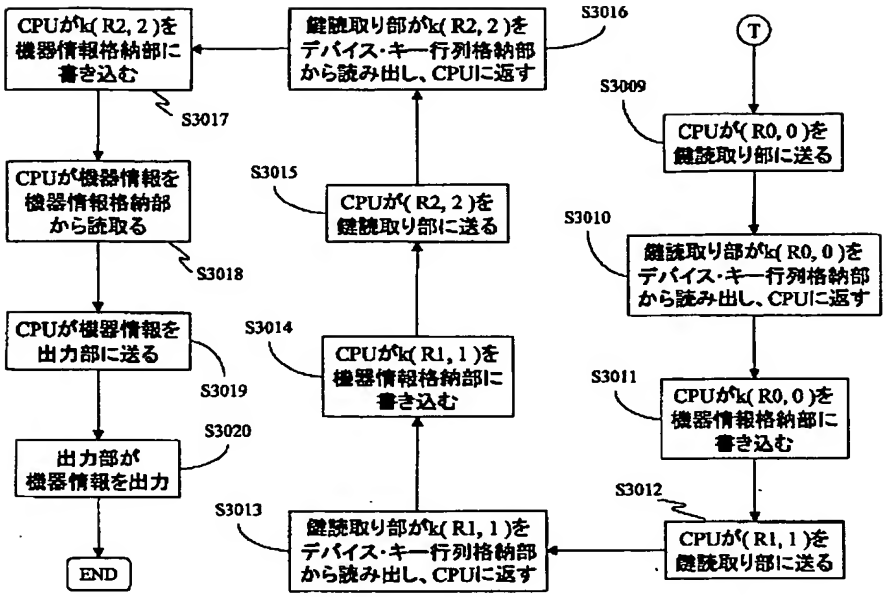
【図４０】


501
入力部
502
CPU
503
作業メモリ
504
乱数発生部
508
鍵読取り部
506
機器情報格納部
507
既出力情報格納部
出力部
505
デバイス・キー行列格納部
509
デバイス情報生成装置
50

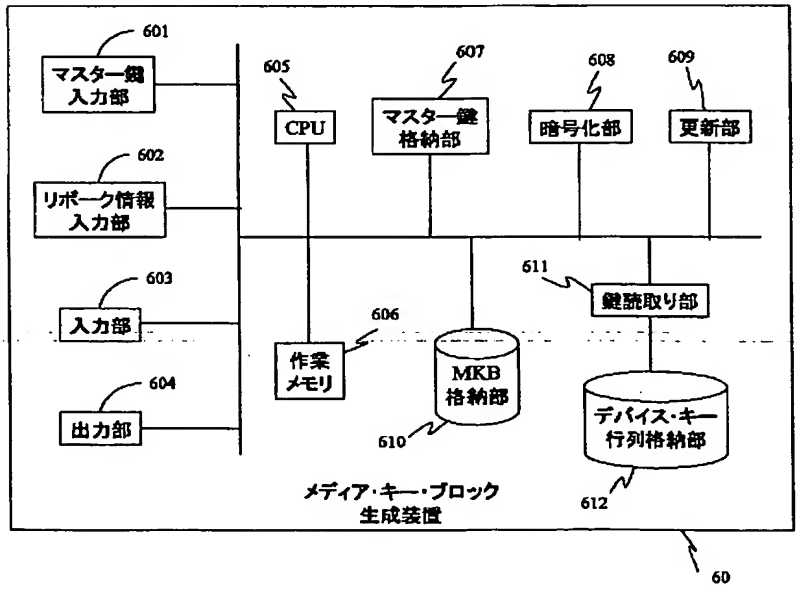
【図４１】


T
START
S3001
S3008
CPUが(R0, R1, R2)を機器情報格納部に格納
入力部が機器情報生成要求を受け取り、当該要求をCPUに送る
S3007
S3002
CPUが(R0, R1, R2)を作業メモリから読み出し、既出力情報格納部に格納
CPUが乱数発生部に乱数発生を指示
S3006
no
S3003
(R0, R1, R2)が見つかった?
yes
乱数発生部が三つの乱数R0, R1, R2を発生し、それらをCPUに送る
S3005
S3004
CPUが数の組(R0, R1, R2)を既出力情報格納部から検索
CPUがR0, R1, R2を作業メモリに格納

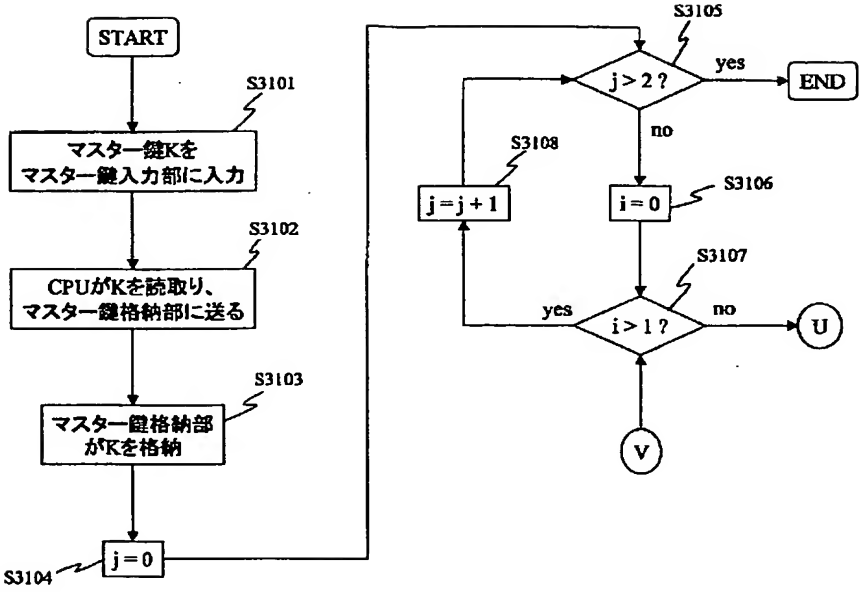
【図４２】


CPUがk(R2,2)を機器情報格納部に書き込む
S3017
CPUが機器情報を機器情報格納部から読取る
S3018
CPUが機器情報を出力部に送る
S3019
出力部が機器情報を出力
S3020
END
鍵読取り部がk(R2,2)をデバイス・キー行列格納部から読み出し、CPUに返す
S3016
CPUが(R2,2)を鍵読取り部に送る
S3015
CPUがk(R1,1)を機器情報格納部に書き込む
S3014
鍵読取り部がk(R1,1)をデバイス・キー行列格納部から読み出し、CPUに返す
S3013
T
CPUが(R0,0)を鍵読取り部に送る
S3009
鍵読取り部がk(R0,0)をデバイス・キー行列格納部から読み出し、CPUに返す
S3010
CPUがk(R0,0)を機器情報格納部に書き込む
S3011
CPUが(R1,1)を鍵読取り部に送る
S3012

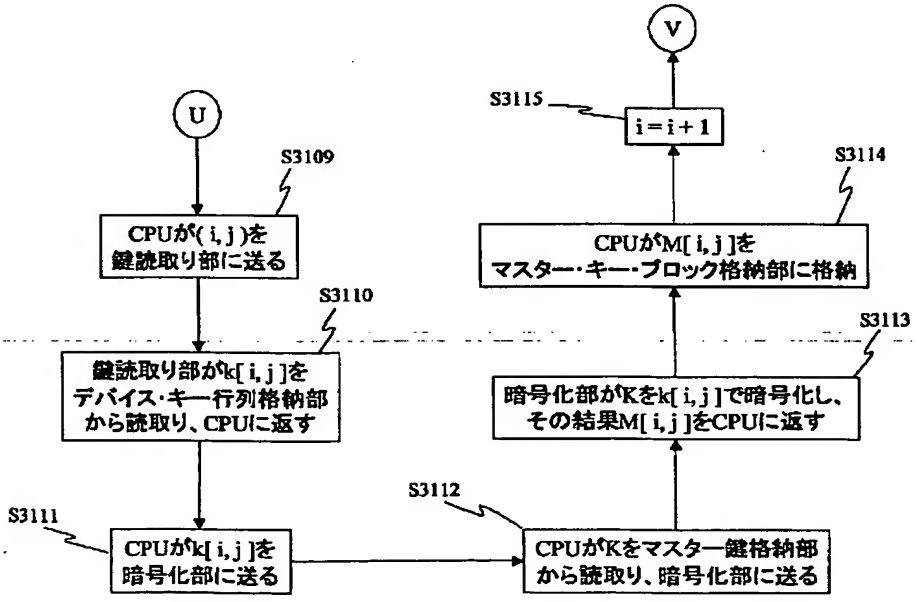
【図４３】


601
マスター鍵入力部
602
リポーク情報入力部
603
入力部
604
出力部
605
CPU
606
作業メモリ
607
マスター鍵格納部
608
暗号化部
609
更新部
610
MKB格納部
611
鍵読取り部
デバイス・キー行列格納部
612
メディア・キー・ブロック生成装置
60

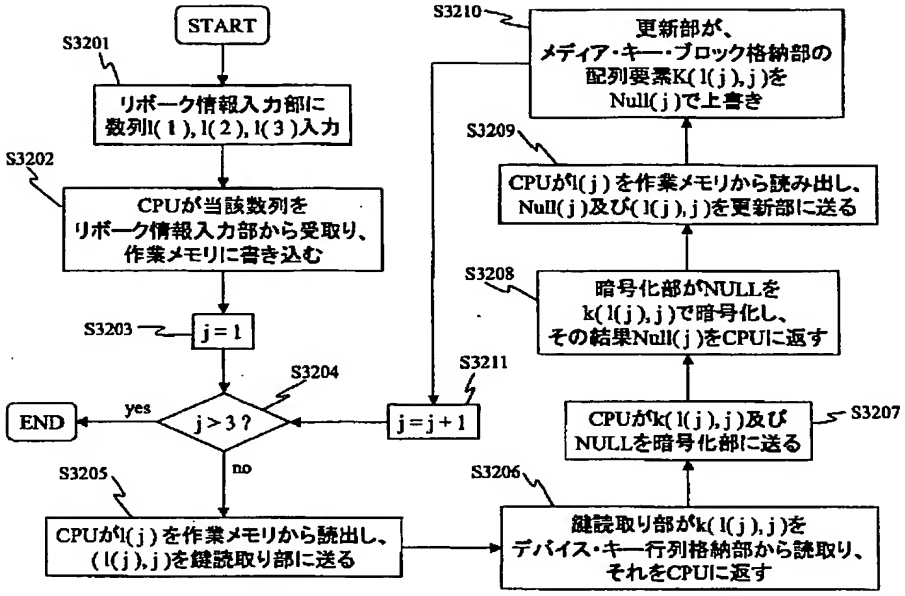
【図４４】


START
S3101
マスター鍵Kを
マスター鍵入力部に入力
S3102
CPUがKを読取り、
マスター鍵格納部に送る
S3103
マスター鍵格納部
がKを格納
S3104
j=0
S3105
j>2?
yes
END
no
S3108
j=j+1
S3106
i=0
S3107
yes
i>1?
no
U
V

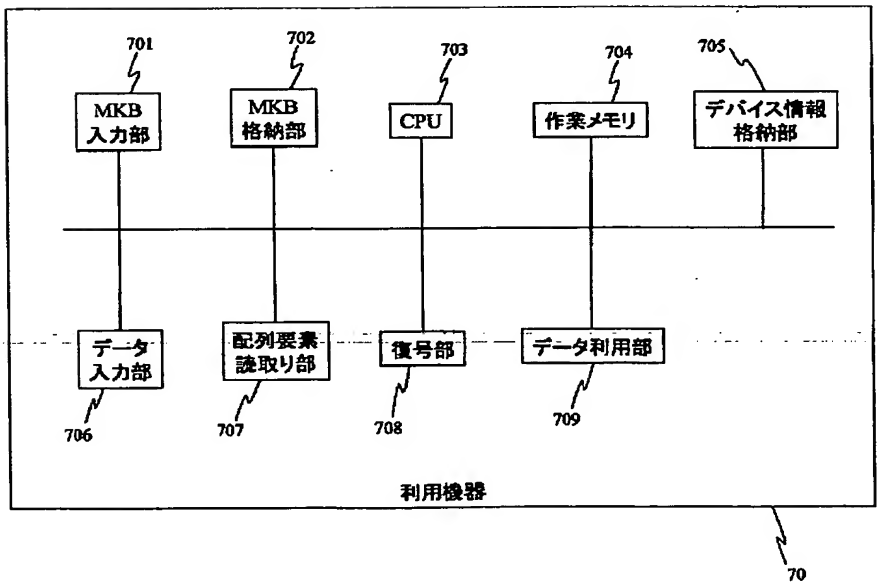
【図４５】


U
S3109
CPUが(i,j)を
鍵読取り部に送る
S3110
鍵読取り部がk[i,j]を
デバイス・キー行列格納部
から読取り、CPUに返す
S3111
CPUがk[i,j]を
暗号化部に送る
S3112
CPUがKをマスター鍵格納部
から読取り、暗号化部に送る
S3113
暗号化部がKをk[i,j]で暗号化し、
その結果M[i,j]をCPUに返す
S3114
CPUがM[i,j]を
マスター・キー・ブロック格納部に格納
S3115
i=i+1
V

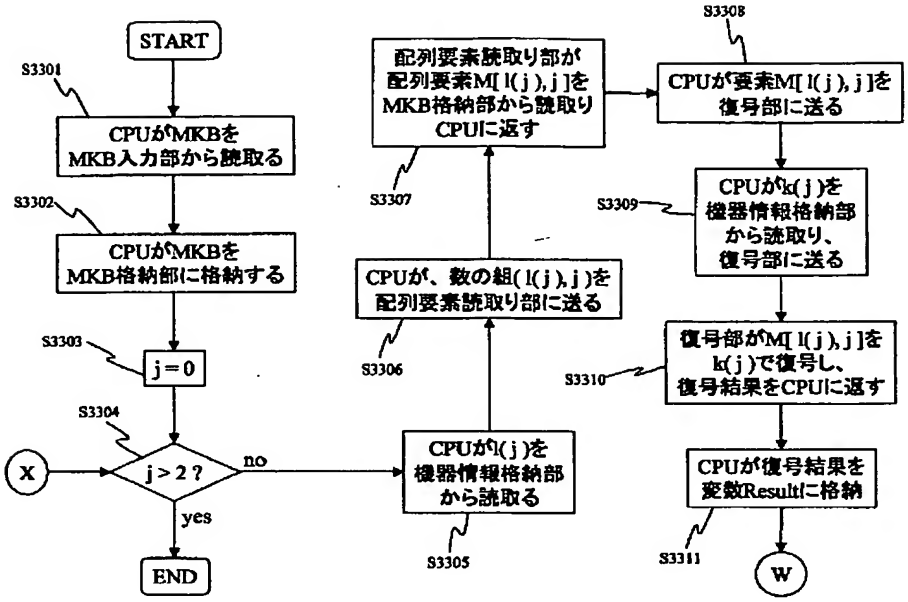
【図４６】


START
S3201
リポーク情報入力部に
数列I(1),I(2),I(3)入力
S3202
CPUが当該数列を
リポーク情報入力部から受取り、
作業メモリに書き込む
S3203
j=1
S3204
j>3?
yes
END
no
S3205
CPUがI(j)を作業メモリから読出し、
(I(j),j)を鍵読取り部に送る
S3206
鍵読取り部がk(I(j),j)を
デバイス・キー行列格納部から読取り、
それをCPUに返す
S3207
CPUがk(I(j),j)及び
NULLを暗号化部に送る
S3208
暗号化部がNULLを
k(I(j),j)で暗号化し、
その結果Null(j)をCPUに返す
S3209
CPUがI(j)を作業メモリから読み出し、
Null(j)及び(I(j),j)を更新部に送る
S3210
更新部が、
メディア・キー・ブロック格納部の
配列要素K(I(j),j)を
Null(j)で上書き
S3211
j=j+1

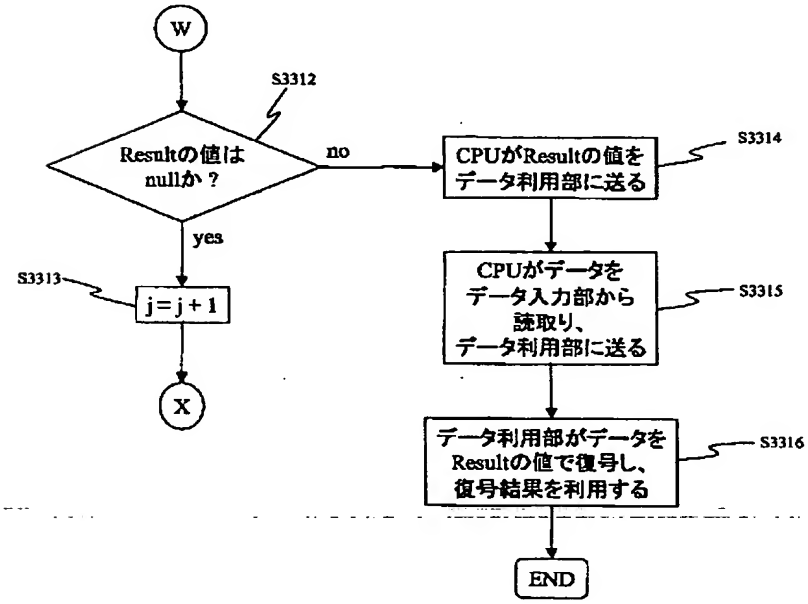
【図４７】


701
MKB
入力部
702
MKB
格納部
703
CPU
704
作業メモリ
705
デバイス情報
格納部
データ
入力部
706
配列要素
読取り部
707
復号部
708
データ利用部
709
利用機器
70

【図４８】


START
S3301
CPUがMKBを
MKB入力部から読取る
S3302
CPUがMKBを
MKB格納部に格納する
S3303
j=0
S3304
X
j>2?
no
yes
END
CPUがl(j)を
機器情報格納部
から読取る
S3305
CPUが、数の組(l(j),j)を
配列要素読取り部に送る
S3306
配列要素読取り部が
配列要素M[l(j),j]を
MKB格納部から読取り
CPUに返す
S3307
S3308
CPUが要素M[l(j),j]を
復号部に送る
S3309
CPUがk(j)を
機器情報格納部
から読取り、
復号部に送る
S3310
復号部がM[l(j),j]を
k(j)で復号し、
復号結果をCPUに返す
CPUが復号結果を
変数Resultに格納
S3311
W

【図４９】


W
S3312
Resultの値は
nullか？
no
yes
S3313
j=j+1
X
CPUがResultの値を
データ利用部に送る
S3314
CPUがデータを
データ入力部から
読取り、
データ利用部に送る
S3315
データ利用部がデータを
Resultの値で復号し、
復号結果を利用する
S3316
END